Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language

W3C Working Draft 10 November 2003

This version:
http://www.w3.org/TR/2003/WD-wsdl20-20031110
Latest version:
http://www.w3.org/TR/wsdl20
Previous versions:
http://www.w3.org/TR/2003/WD-wsdl12-20030611

Editors:
Roberto Chinnici, Sun Microsystems
Martin Gudgin, Microsoft
Jean-Jacques Moreau, Canon
Jeffrey Schlimmer, Microsoft
Sanjiva Weerawarana, IBM Research

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

Abstract

This document describes the Web Services Description Language (WSDL) Version 2.0, an XML language for describing Web services. This specification defines the core language which can be used to describe Web services based on an abstract model of what the service offers.

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 Working Draft of the Web Services Description Language (WSDL) 2.0 document.

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.

Comments on this document are invited and are to be sent to the public www-ws-desc@w3.org mailing list (public archive).

Patent disclosures relevant to this specification may be found on the Working Group's patent disclosure page.

Short Table of Contents

Table of Contents

Appendices

---

1. Introduction

Web Services Description Language (WSDL) provides a model and an XML format for describing Web services. WSDL enables one to separate the description of the abstract functionality offered by a service from concrete details of a service description such as "how" and "where" that functionality is offered.

This specification defines a language for describing the abstract functionality of a service as well as a framework for describing the concrete details of a service description. The *WSDL Version 2.0 Part 2: Message Exchange Patterns* specification [*WSDL 2.0 Message Exchange Patterns [p.58]*] defines define the sequence and cardinality of abstract messages sent or received by an operation. The *WSDL Version 2.0 Part 3: Bindings* specification [*WSDL 2.0 Bindings [p.58]*] defines a language for describing such concrete details for SOAP 1.2 [*SOAP 1.2 Part 1: Messaging Framework [p.59]*], HTTP [*IETF RFC 2616 [p.59]*] and MIME [*IETF RFC 2045 [p.58]*].

1.1 Web Service

WSDL describes a Web service in two fundamental stages: one abstract and one concrete. Within each stage, the description uses a number of constructs to promote reusability of the description and separate independent design concerns.

At an abstract level, WSDL describes a Web service in terms of the messages it sends and receives; messages are described independent of a specific wire format using a type system, typically XML Schema.

An *operation* associates a message exchange pattern with one or more messages. A *message exchange pattern* identifies the sequence and cardinality of messages sent and/or received as well as who they are logically sent to and/or received from. An *interface* groups together operations without any commitment to transport or wire format.

At a concrete level, a *binding* specifies transport and wire format details for one or more interfaces. An *endpoint* associates a network address with a binding. And finally, a *service* groups together endpoints that implement a common interface.

1.2 Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [*IETF RFC 2119 [p.57]*].

This specification uses properties from the XML Information Set [*XML Information Set [p.57]*]. Such properties are denoted by square brackets, e.g. [namespace name].

This specification uses namespace prefixes throughout; they are listed in Table 1-1 [p.7] . Note that the choice of any namespace prefix is arbitrary and not semantically significant (see [*XML Information Set [p.57]*]).

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

Prefix	Namespace	Notes
wsdl	"http://www.w3.org/2003/11/wsdl"	A normative XML Schema [*XML Schema: Structures [p.58]*], [*XML Schema: Datatypes [p.58]*] document for the "http://www.w3.org/2003/11/wsdl" namespace can be found at http://www.w3.org/2003/11/wsdl. WSDL documents that do NOT conform to this schema are not valid WSDL documents. WSDL documents that DO conform to this schema and also conform to the other constraints defined in this specification are valid WSDL documents.
wsoap12	"http://www.w3.org/2003/11/wsdl/soap12"	Defined by WSDL 2.0: Bindings [*WSDL 2.0 Bindings [p.58]*].
whttp	"http://www.w3.org/2003/11/wsdl/http"	
wmime	"http://www.w3.org/2003/11/wsdl/mime"	
xs	"http://www.w3.org/2001/XMLSchema"	Defined in the W3C XML Schema specification [*XML Schema: Structures [p.58]*], [*XML Schema: Datatypes [p.58]*].
xsi	"http://www.w3.org/2001/XMLSchema-instance"	

Namespace names of the general form "http://example.org/..." and "http://example.com/..." represent application or context-dependent URIs [*IETF RFC 2396 [p.57]*].

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.

2. Component Model

This section describes the conceptual model for WSDL as a set of components with properties, each aspect of a Web service that WSDL can describe having its own property. In addition an XML Infoset representation for these components is provided, along with a mapping from that representation to the various component properties. How the XML Infoset representation of a given set of WSDL components is constructed is outside the scope of this specification.

2.1 Definitions

2.1.1 The Definitions Component

At the abstract level, the Definitions component is just a container for two categories of component; WSDL components and type system components.

WSDL components are interfaces, bindings and services.

Type system components are element declarations and type definitions drawn from some type system. The former define the [local name], [namespace name], [children] and [attributes] properties of an *element information item*; the latter define only the [children] and [attributes] properties.

The properties of the Definitions component are as follows:

- {interfaces} A set of named interface definitions
- {bindings} A set of named binding definitions
- {services} A set of named service definitions
- {type definitions} A set of named type definitions, each one isomorphic to a simple or complex type as defined by XML Schema
- {element declarations} A set of named element declarations, each one isomorphic to a global element declaration as defined by XML Schema

The set of interfaces/binding/services/etc. available in the Definitions component include those that are defined within the component itself and those that are imported and/or included. At the component model level, there is no distinction between directly defined components vs. imported/included components.

The components directly defined within a single Definitions component are said to belong to the same *target namespace*. The target namespace therefore groups a set of related component definitions and provides a hint of the intended semantics of the components.

Imported components have different target namespace values from the Definitions component that is importing them. Thus importing is the mechanism to use components from one namespace in another set of definitions.

Each WSDL or type/element component MUST be uniquely identified by its qualified name. That is, if two distinct components of the same kind (Interface, Binding etc.) are in the same target namespace, then their QNames MUST be unique. However, different kids of components (e.g., an Interface component and a Binding component) MAY have the same QName. Thus, QNames of components must be unique within the space of those components in a given target namespace.

In addition to WSDL components and type and element components, additional extension components MAY be added via extensibility **6. Language Extensibility** [p.56] . Further, additional properties to WSDL and type/element components MAY also be added via extensibility.

2.1.2 XML Representation of Definitions Component

```
<definitions
      targetNamespace="xs:anyURI" >
  <documentation />?
  [ <import /> | <include /> ]*
  <types />?
  [ <interface /> | <binding /> | <service /> ]*
</definitions>
```

WSDL definitions are represented in XML by one or more WSDL Information Sets (Infosets), that is one or more `definitions` *element information item*s. A WSDL Infoset contains representations for a collection of WSDL components which share a common target namespace. A WSDL Infoset which contains one or more `import` *element information item*s **4.2 Importing Descriptions** [p.53] corresponds to a collection with components drawn from multiple target namespaces.

The target namespace represents an unambiguous name for the intended semantics of the WSDL Infoset. The targetNamespace URI SHOULD point to a human or machine processable document that directly or indirectly defines the semantics of the WSDL Infoset.

The `definitions` *element information item* has the following Infoset properties:

- A [local name] of `definitions` .
- A [namespace name] of "http://www.w3.org/2003/11/wsdl".
- One or more *attribute information item*s amongst its [attributes] as follows:
  - A REQUIRED `targetNamespace` *attribute information item* as described below in **2.1.2.1 targetNamespace attribute information item** [p.10] .
  - 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/2003/11/wsdl".
- Zero or more *element information item*s amongst its [children], in order as follows:
  1. An OPTIONAL `documentation` *element information item* (see **5. Documentation** [p.55]).
  2. Zero or more *element information item*s from among the following, in any order:
     - Zero or more `include` *element information item*s (see **4.1 Including Descriptions** [p.52])
     - Zero or more `import` *element information item*s (see **4.2 Importing Descriptions** [p.53])
     - Zero or more namespace-qualified *element information item*s. The [namespace name] of such *element information item*s MUST NOT be "http://www.w3.org/2003/11/wsdl".

3. An OPTIONAL `types` *element information item* (see **3. Types** [p.48]).
4. Zero or more *element information item*s from among the following, in any order:
   - `interface` *element information item*s (see **2.2.2 XML Representation of Interface Component** [p.12]).
   - `binding` *element information item*s (see **2.8.2 XML Representation of Binding Component** [p.33]).
   - `service` *element information item*s (see **2.12.2 XML Representation of Service Component** [p.43]).
   - Zero or more namespace-qualified *element information item*s. The [namespace name] of such *element information item*s MUST NOT be "http://www.w3.org/2003/11/wsdl".

2.1.2.1 `targetNamespace` *attribute information item*

The `targetNamespace` *attribute information item* defines the namespace affiliation of top-level components defined in this `definitions` *element information item*. Interfaces, Bindings and Services are top-level components.

The `targetNamespace` *attribute information item* has the following Infoset properties:

- A [local name] of `targetNamespace`
- A [namespace name] which has no value

The type of the `targetNamespace` *attribute information item* is *xs:anyURI*.

2.1.3 Mapping Definitions' XML Representation to Component Properties

The mapping between the properties of the Definitions component (see **2.1.1 The Definitions Component** [p.8]) and the XML Representation of the `definitions` *element information item* (see **2.1.2 XML Representation of Definitions Component** [p.9]) is described in Table 2-1 [p.10] .

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

Property	Mapping
{interfaces}	The interface definitions corresponding to all the `interface` *element information item*s in the [children] of the `definitions` *element information item*, if any, plus any included or imported interface definitions (see **4. Modularizing WSDL descriptions** [p.52]).
{bindings}	The binding definitions corresponding to all the `binding` *element information item*s in the [children] of the `definitions` *element information item*, if any, plus any included or imported binding definitions (see **4. Modularizing WSDL descriptions** [p.52]).
{services}	The service definitions corresponding to all the `service` *element information item*s in the [children] of the `definitions` *element information item*, if any, plus any included or imported service definitions (see **4. Modularizing WSDL descriptions** [p.52]).
{type definitions}	The type definition components corresponding to all the type definitions defined as descendants of the `types` *element information item*, if any, plus any imported type definitions. At a minimum this will include all the types defined by XML Schema `simpleType` and `complexType` *element information item*s. It MAY also include any definition from some other type system which describes the [attributes] and [children] properties of an *element information item*.
{element declarations}	The element declaration components corresponding to all the element declarations defined as descendants of the `types` *element information item*, if any, plus any imported element definitions. At a minimum this will include all the global element declarations defined by XML Schema `element` *element information item*s. It MAY also include any definition from some other type system which describes the [local name], [namespace name], [attributes] and [children] properties of an *element information item*.

2.2 Interface

2.2.1 The Interface Component

An Interface component describes sets of messages that a service sends and/or receives. It does this by grouping related messages into operations. An operation is a set of input and output messages, an interface is a set of operations.

An interface can optionally extend one or more other interfaces. In such cases the interface contains the operations of the interfaces it extends, along with any operations it defines. The interfaces a given interface extends MUST NOT themselves extend that interface either directly or indirectly.

Interfaces are named constructs and can be referred to by QName (see **2.16 QName resolution** [p.48]). For instance, Binding components refer to interfaces in this way.

The properties of the Interface component are as follows:

- {name} An NCName as defined by [*XML Namespaces [p.58]*].
- {target namespace} A namespace name, as defined in [*XML Namespaces [p.58]*].
- {extended interfaces} A set of named interface definitions which this interface extends.
- {style default} The default style of message schemas of the operations. If a per-operation value is not specified for this property, then this value becomes the default value of the property. See **2.3.2.3 style attribute information item with operation [owner]** [p.20]
- {operations} A set of named interface operation definitions.
- {features} A set of named feature definitions.
- {properties} A set of named property definitions.

For each Interface component in the {interfaces} property of a definitions container the combination of {name} and {target namespace} properties must be unique.

2.2.2 XML Representation of Interface Component

```
<definitions>
  <interface
        name="xs:NCName"
        extends="list of xs:QName"?
        styleDefault="xs:anyURI"? >
    <documentation />?
    [ <operation /> | <feature /> | <property /> ]*
  </interface>
</definitions>
```

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

- A [local name] of `interface`
- A [namespace name] of "http://www.w3.org/2003/11/wsdl"
- One or more *attribute information item*s amongst its [attributes] as follows:
    - A REQUIRED `name` *attribute information item* as described below in **2.2.2.1 name attribute information item with interface [owner]** [p.13] .
    - An OPTIONAL `extends` *attribute information item* as described below in **2.2.2.2 extends attribute information item** [p.13] .

    - An OPTIONAL `styleDefault` *attribute information item* as described below in **2.2.2.3 styleDefault attribute information item** [p.14] .
    - Zero or more namespace qualified *attribute information items*. The [namespace name] of such *attribute information items* MUST NOT be "http://www.w3.org/2003/11/wsdl".
- Zero or more *element information items* amongst its [children], in order, as follows:
    1. An OPTIONAL `documentation` *element information item* (see **5. Documentation** [p.55]).
    2. Zero or more *element information items* from among the following, in any order:
        - Zero or more `operation` *element information items* **2.3.2 XML Representation of Interface Operation Component** [p.18] .
        - Zero or more `feature` *element information items* **2.6.2 XML Representation of Feature Component** [p.28] .
        - Zero or more `property` *element information items* **2.7.2 XML Representation of Property Component** [p.30] .
        - Zero or more namespace-qualified *element information items* amongst its [children]. The [namespace name] of such *element information items* MUST NOT be "http://www.w3.org/2003/11/wsdl".

2.2.2.1 `name` *attribute information item* with `interface` [owner]

The `name` *attribute information item* together with the `targetNamespace` *attribute information item* of the `definitions` *element information item* forms the QName of the interface.

The `name` *attribute information item* has the following Infoset properties:

- A [local name] of `name`
- A [namespace name] which has no value

The type of the `name` *attribute information item* is *xs:NCName*.

2.2.2.2 `extends` *attribute information item*

The `extends` *attribute information item* lists the interfaces that this interface derives from.

The `extends` *attribute information item* has the following Infoset properties:

- A [local name] of `extends`
- A [namespace name] which has no value

The type of the `extends` *attribute information item* is a list of *xs:QName*.

2.2.2.3 `styleDefault` *attribute information item*

The `styleDefault` *attribute information item* indicates the default style used to construct the {message} properties of {message references} of all operations contained within the [owner] `interface` .

The `styleDefault` *attribute information item* has the following Infoset properties:

- A [local name] of `styleDefault.`
- A [namespace name] which has no value.

The type of the `styleDefault` *attribute information item* is *xs:anyURI*.

2.2.3 Mapping Interface's XML Representation to Component Properties

The mapping between the properties of the Interface component (see **2.2.1 The Interface Component** [p.11]) and the XML Representation of the `interface` *element information item* (see **2.2.2 XML Representation of Interface Component** [p.12]) is as described in Table 2-2 [p.14] .

Table 2-2. Mapping between Interface Component Properties and XML Representation

Property	Mapping
{name}	The actual value of the `name` *attribute information item*
{target namespace}	The actual value of the `targetNamespace` *attribute information item* of the [parent] `definitions` *element information item*
{style default}	The actual value of the `styleDefault` *attribute information item*
{extended interfaces}	The set of interface definitions resolved to by the values in the `extends` *attribute information item* if any, plus the set of interface definitions in the {extended interfaces} property of those interface definitions, otherwise empty.
{operations}	The set of interface operation definitions corresponding to the `operation` *element information item*s in [children], if any, plus the set of interface operation definitions in the {operations} property of the interface definitions in {extended interfaces}, if any.
{features}	The set of feature definitions corresponding to the `feature` *element information item*s in [children], if any, plus the set of feature definitions in the {features} property of the feature definitions in {extended interfaces}, if any.
{properties}	The set of property definitions corresponding to the `property` *element information item*s in [children], if any, plus the set of property definitions in the {properties} property of the property definitions in {extended interfaces}, if any.

Note:

Per **2.2.1 The Interface Component** [p.11] , the Interface components in the {extended interfaces} property of a given Interface component MUST NOT contain that Interface component in any of their {extended interfaces} properties, that is to say, recursive extension of interfaces is disallowed.

2.3 Interface Operation

2.3.1 The Interface Operation Component

An Interface Operation component describes an operation that a given interface supports. An operation is an interaction with the service consisting of a set (ordinary and fault) messages exchanged between the service and the other roles involved in the interaction, in particular the service requester. The sequencing and cardinality of the messages involved in a particular interaction is governed by the *message exchange pattern* used by the operation (see {message exchange pattern} property).

A message exchange pattern defines placeholders for messages, the participants in the pattern (i.e., the sources and sinks of the messages), and the cardinality and sequencing of messages exchanged by the participants. The message placeholders are associated with specific message types by the operation using the pattern via message and fault references (see {message references} and {fault references} properties). The service whose operation is using the pattern becomes one of the participants of the pattern. This specification does not define a machine understandable language for defining message exchange patterns nor does it define any specific patterns. The companion specification, [*WSDL 2.0 Message Exchange Patterns [p.58]*] defines a set of such patterns and defines identifying URIs any of which MAY be used as the value of the {message exchange pattern} property.

The properties of the Interface Operation component are as follows:

- {name} An NCName as defined by [*XML Namespaces [p.58]*].
- {target namespace} A namespace name, as defined in [*XML Namespaces [p.58]*].
- {message exchange pattern} A URI identifying the message exchange pattern used by the operation.
- {message references} A set of Message Reference components for the ordinary messages the operation accepts or sends. (See **2.4 Message Reference** [p.21] .)
- {fault references} A set of Fault Reference components for the fault messages the operation accepts or sends. (See **2.5 Fault Reference** [p.24] .)
- {style} A URI identifying the rules that were used to construct the {message} properties of {message references}. (See **2.3.1.1 Operation Style** [p.16]
- {features} A set of named feature definitions used by the operation
- {properties} A set of named property definitions used by the operation

Interface Operation components are local to Interface components; they cannot be referred to by QName, despite having both {name} and {target namespace} properties. That is, this specification does not preclude one from having two distinct Interface Operation components which have the same {name} and {target namespace} properties, as long as they are in different Interface components (with the same {target namespace} property).

For each Interface Operation component in the {operations} property of an Interface component the combination of {name} and {target namespace} properties must be unique.

In cases where, due to an interface extending one or more other interfaces, two or more Interface Operation components have the same value for their {name} and {target namespace} properties, then the component models of those Interface Operation components MUST be equivalent (see **2.14 Equivalence of Components** [p.47]). If the Interface Operation components are equivalent then they are considered to collapse into a single component. It is an error if two Interface Operation components have the same value for their {name} and {target namespace} properties but are not equivalent.

Note:

Due to the above rules, if two interfaces that have the same value for their {target namespace} property also have one or more operations that have the same value for their {name} property then those two interfaces cannot both form part of the derivation chain of a derived interface unless those operations are the same operation. Therefore it is considered good practice to ensure, where necessary, that the {name} property of Interface Operation components within a namespace are unique, thus allowing such derivation to occur without inadvertent error.

2.3.1.1 Operation Style

If the {style} property of an Interface Operation component has a value then that value (a URI) implies the rules that were used to define the {message} properties of *all* {message reference}s within that component. Note that the property MAY not have any value. If this property has a given value, then the rules implied by that value (such as rules that govern the schemas) MUST be followed or it is an error.

This specification defines the following pre-defined operation styles:

- RPC Style (see **2.3.1.1.1 RPC Style** [p.16])
- Set-Attribute Style (see **2.3.1.1.2 Set-Attribute Style** [p.18])
- Get-Attribute Style (see **2.3.1.1.3 Get-Attribute Style** [p.18])

2.3.1.1.1 RPC Style

The RPC style is selected by assigning the Interface Operation component’s {style} property the value *http://www.w3.org/2003/11/wsdl/style/rpc*.

The RPC style may only be used for Interface Operation components whose {message exchange pattern} property has the value ’http://www.w3.org/2003/11/wsdl/in-only’ or ’http://www.w3.org/2003/11/wsdl/in-out’.

Use of this value indicates that XML Schema [*XML Schema: Structures [p.58]*] was used to define the schemas of the {message} properties of all {message reference} components of the Interface Operation component. Those schemas MUST adhere to the rules below.

Note that if the Interface Operation component uses the {message exchange pattern} ’http://www.w3.org/2003/11/wsdl/in-only’ then there is no output element and hence the rules which refer to the output element do not apply.

- The content model of input and output {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. Note that these child elements MAY contain the following attributes: nillable, minOccurs and maxOccurs.
- The localPart of input element’s QName MUST be the same as the Interface operation component’s name.
- The localPart of the output element’s QName is obtained by concatenating the name of the operation and the string value "Response", i.e. concat(operation/@name,"Response").
- Input and output elements MUST both be in the same namespace.
- The complex type that defines the body of an input or an output element MUST NOT contain any attributes.
- If elements with the same qualified name appear as children of both the input and output elements, then they MUST both be declared using the same type.
- The input or output sequence MUST NOT contain multiple children element declared with the same name.

Furthermore, the following rules MAY BE used to map between a message and a signature of a remote procedure call. These rules are suggested conventions to accommodate mapping.

- If an element is a child of the input element and an element with the same name is not a child of the output element, then it represents an input parameter.
- If an element is a child of the output element and an element with the same name is not a child of the input element, then it represents an output parameter.
- If an element that occurs as a child element both in the input and the output elements, then it represents an in/out parameter.
- Each element represents a single parameter irrespective of its cardinality.

Editorial note: SW	20031103
The WG has adopted a proposal (http://lists.w3.org/Archives/Public/www-ws-desc/2003Oct/0347.html) to capture the signature using an extension attribute - that will be included here at a later draft.	

2.3.1.1.2 Set-Attribute Style

The Set-Attribute style is selected by assigning the Interface Operation component’s {style} property the value *http://www.w3.org/2003/11/wsdl/style/set-attribute*.

Editorial note: JS	20031105
The WG has adopted the rules recommended by the attributes task force for this style, and these rules will be included in an updated draft.	

2.3.1.1.3 Get-Attribute Style

The Get-Attribute style is selected by assigning the Interface Operation component’s {style} property the value *http://www.w3.org/2003/11/wsdl/style/get-attribute*.

Editorial note: JS	20031105
The WG has adopted the rules recommended by the attributes task force for this style, and these rules will be included in an updated draft.	

2.3.2 XML Representation of Interface Operation Component

```
<definitions>
  <interface>
    <operation
          name="xs:NCName"
          pattern="xs:anyURI"
          style="xs:anyURI"? >
      <documentation />?
      [ <feature /> | <property /> |
        [ <input /> | <output /> | <infault /> | <outfault /> ]+
      ]*
    </operation>
  </interface>
</definitions>
```

The XML representation for an Interface Operation component is an *element information item* with the following Infoset properties:

- A [local name] of `operation`
- A [namespace name] of "http://www.w3.org/2003/11/wsdl"

- Two or more *attribute information item*s amongst its [attributes] as follows:
  - A REQUIRED `name` *attribute information item* as described below in **2.3.2.1 name attribute information item with operation [owner]** [p.20] .
  - A REQUIRED `pattern` *attribute information item* as described below in **2.3.2.2 pattern attribute information item with operation [owner]** [p.20] .
  - An OPTIONAL `style` *attribute information item* as described below in **2.3.2.3 style attribute information item with operation [owner]** [p.20] .
  - 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/2003/11/wsdl".
- Zero or more *element information item* amongst its [children], in order, as follows:
  1. An OPTIONAL `documentation` *element information item* (see **5. Documentation** [p.55]).
  2. Zero or more *element information item*s from among the following, in any order:
     - Zero or more `input` *element information item*s (see **2.4.2 XML Representation of Message Reference Component** [p.22]).
     - Zero or more `output` *element information item*s (see **2.4.2 XML Representation of Message Reference Component** [p.22]).
     - Zero or more `infault` *element information item*s (see **2.5.2 XML Representation of Fault Reference Component** [p.25]).
     - Zero or more `outfault` *element information item*s (see **2.5.2 XML Representation of Fault Reference Component** [p.25]).
     - A `feature` *element information item* (see **2.6.2 XML Representation of Feature Component** [p.28]).
     - A `property` *element information item* (see **2.7.2 XML Representation of Property Component** [p.30]).
     - 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/2003/11/wsdl".
- At least one of the [children] MUST be an `input` , `output` , `infault` , or `outfault` *element information item*.

2.3.2.1 `name` *attribute information item* with `operation` [owner]

The `name` *attribute information item* identifies a given `operation` *element information item* inside a given `interface` *element information item*.

The `name` *attribute information item* has the following Infoset properties:

- A [local name] of `name`
- A [namespace name] which has no value

The type of the `name` *attribute information item* is *xs:NCName*.

2.3.2.2 `pattern` *attribute information item* with `operation` [owner]

The `pattern` *attribute information item* identifies the message exchange pattern a given operation uses.

The `pattern` *attribute information item* has the following Infoset properties:

- A [local name] of `pattern`
- A [namespace name] which has no value

The type of the `pattern` *attribute information item* is *xs:anyURI*.

2.3.2.3 `style` *attribute information item* with `operation` [owner]

The `style` *attribute information item* indicates the rules that were used to construct the {message} properties of {message reference} components of the [owner] operation.

The `style` *attribute information item* has the following Infoset properties:

- A [local name] of `style`
- A [namespace name] which has no value

The type of the `style` *attribute information item* is *xs:anyURI*.

2.3.3 Mapping Interface Operation’s XML Representation to Component Properties

The mapping between the properties of the Interface Operation component (see **2.3.1 The Interface Operation Component** [p.15]) and the XML Representation of the `interface` *element information item* (see **2.3.2 XML Representation of Interface Operation Component** [p.18]) is as described in Table 2-3 [p.20] .

Table 2-3. Mapping between Interface Operation Component Properties and XML Representation

Property	Mapping
{name}	The actual value of the `name` *attribute information item*
{target namespace}	The actual value of the `targetNamespace` *attribute information item* of the [parent] `definitions` *element information item* of the [parent] `interface` *element information item*.
{message exchange pattern}	The actual value of the `pattern` *attribute information item*
{message references}	The set of message references corresponding to the `input` and `output` *element information item*s in [children], if any.
{fault references}	The set of fault references corresponding to the `infault` and `outfault` *element information item*s in [children] if any.
{style}	The actual value of the `style` *attribute information item* if present, otherwise the actual value of the `styleDefault` *attribute information item* of the [parent] `interface` *element information item* if present, otherwise none.
{features}	The set of features corresponding to the `feature` *element information item*s in [children], if any.
{properties}	The set of properties corresponding to the `property` *element information item*s in [children], if any.

2.4 Message Reference

2.4.1 The Message Reference Component

A Message Reference component associates XML element declarations that define the message content for one of the messages participating in an operation.

Message Reference components are identified by the role the message plays in the {message exchange pattern} that the operation is using. That is, a message exchange pattern defines a set of placeholder messages that participate in the pattern and assigns them unique names within the pattern. The purpose of a Message Reference component is to associate an actual message type (XML element declaration for message content) for the message that will perform a specific role in the message exchange pattern.

The properties of the Message Reference component are as follows:

- {message reference} An NCName as defined by [*XML Namespaces [p.58]*]. This property identifies the role this message plays in the {message exchange pattern} of the Interface Operation component this is contained within. The value of this property MUST match the name of a placeholder message defined by the message exchange pattern.

- {direction} One of *in* or *out* indicating whether the message is coming to the service or going from the service, respectively. The direction MUST be the same as the direction of the message identified by the {message reference} property in the {message exchange pattern} of the Interface Operation component this is contained within.

- {message} A reference to an XML element declaration. This element represents the content or "payload" of the message.

For each Message Reference component in the {message references} property of an Interface Operation component the {message reference} property MUST be unique.

2.4.2 XML Representation of Message Reference Component

```
<definitions>
  <interface>
    <operation>
      <input
            messageReference="xs:NCName"?
            message="xs:QName"? >
        <documentation />?
      </input>
      <output
            messageReference="xs:NCName"?
            message="xs:QName"? >
        <documentation />?
      </output>
    </operation>
  </interface>
</definitions>
```

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

- A [local name] of `input` or `output`

- A [namespace name] of "http://www.w3.org/2003/11/wsdl"

- Zero or more *attribute information item*s amongst its [attributes] as follows:

  - An OPTIONAL `messageReference` *attribute information item* as described below in **2.4.2.1 messageReference attribute information item with input, or output [owner]** [p.23] .

    If the {message exchange pattern} of the Interface Operation component has only one message with a given value for {direction}, then the `messageReference` *attribute information item* is optional for the XML representation of the Message Reference component with that {direction}.

  - An OPTIONAL `message` *attribute information item* as described below in **2.4.2.2 message attribute information item with input, or output [owner]** [p.23] .

    - 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/2003/11/wsdl".

- Zero or more *element information item*s amongst its [children], in order, as follows:

    1. An OPTIONAL `documentation` *element information item* (see **5. Documentation** [p.55]).
    2. Zero or more namespace-qualified *element information item*s. The [namespace name] of such *element information item*s MUST NOT be "http://www.w3.org/2003/11/wsdl".

2.4.2.1 `messageReference` *attribute information item* with `input` , or `output` [owner]

The `messageReference` *attribute information item* identifies the role of this message in the message exchange pattern of the given `operation` *element information item*.

The `messageReference` *attribute information item* has the following Infoset properties:

- A [local name] of `messageReference`
- A [namespace name] which has no value

The type of the `messageReference` *attribute information item* is *xs:NCName*.

2.4.2.2 `message` *attribute information item* with `input` , or `output` [owner]

The `message` *attribute information item* refers, by QName, to an element declaration component.

The `message` *attribute information item* has the following Infoset properties:

- A [local name] of `message` .
- A [namespace name] which has no value.

The type of the `message` *attribute information item* is *xs:QName*.

2.4.3 Mapping Message Reference’s XML Representation to Component Properties

The mapping between the properties of the Message Reference component (see **2.4.1 The Message Reference Component** [p.21]) and the XML Representation of the message reference *element information item* (see **2.4.2 XML Representation of Message Reference Component** [p.22]) is as described in Table 2-4 [p.23] .

Table 2-4. Mapping between Message Reference Component Properties and XML Representation

Property	Mapping
{message reference}	The actual value of the `messageReference` *attribute information item* if any; otherwise the {message reference} property of the message with same {direction} from the {message exchange pattern} of the Interface Operation component, provided there is exactly one such message; otherwise empty.
{direction}	If the [local name] of the *element information item* is `input` then "in", else if the [local name] of the *element information item* is `output` then "out".
{message}	The element declaration resolved to by the value of the `message` *attribute information item* if present, otherwise a similar construct in some type system as referred to by some other *attribute information item* if present, otherwise empty.

2.5 Fault Reference

2.5.1 The Fault Reference Component

A Fault Reference component associates an XML element declaration that defines the fault message contents for a fault that occurs related to a message participating in an operation.

Fault Reference components are identified by the role the related message plays in the {message exchange pattern} that the operation is using. That is, a message exchange pattern defines a set of placeholder messages that participate in the pattern and assigns them unique names within the pattern. The purpose of a Fault Reference component is to associate an actual message type (XML element declaration) for the fault that will occur related to a specific message in the message exchange pattern.

The companion specification [*WSDL 2.0 Message Exchange Patterns [p.58]*] defines two *fault patterns* that a given message exchange pattern may use. For the pattern *fault-replaces-message*, the message that the fault relates to identifies the message *in place of which* the declared fault message will occur. Thus, the fault message will travel in the *same* direction as the message it replaces in the pattern. For the pattern *message-triggers-fault*, the message that the fault relates to identifies the message after which the indicated fault may occur, in the opposite direction of the referred to message. That is, the fault message will travel in the *opposite* direction of the message it comes after in the pattern.

More than one Fault Reference component may refer to the same message role. This allows one to indicate that there is more than one type of fault that is related to that message.

The properties of the Fault Reference component are as follows:

- {name} An NCName as defined by [*XML Namespaces [p.58]*]. This property is used to identify the fault.

- {message reference} An NCName as defined by [*XML Namespaces [p.58]*]. This property identifies the message which this fault relates to in the {message exchange pattern} of the Interface Operation component this is contained within. The value of this property MUST match the name of a placeholder message defined by the message exchange pattern.

- {direction} One of *in* or *out* indicating whether the fault is coming to the service or going from the service, respectively. The direction MUST be consistent with the direction implied by the fault rule used in the message exchange pattern of the operation. For example, if the fault rule fault-replaces-message is used, then a fault which refers to an outgoing message would have a {direction} property value of *out*. If the fault rule message-triggers-fault is used, then a fault which refers to an outgoing message would have a {direction} property value of *in* as the fault travels in the opposite direction of the message.

- {message} A reference to an XML element declaration. This element represents the content or "payload" of the fault.

If two or more Interface Operation components in the same Interface component have Fault Reference components with the same {name} value, then their {message} properties MUST have the same value. This allows two or more operations to indicate that they both produce the same fault.

2.5.2 XML Representation of Fault Reference Component

```
<definitions>
  <interface>
    <operation>
      <infault
            name="xs:NCName"
            messageReference="xs:NCName"?
            message="xs:QName"?
        <documentation />?
      </infault>*
      <outfault
            name="xs:NCName"
            messageReference="xs:NCName"?
            message="xs:QName"?
        <documentation />?
      </outfault>*
    </operation>
  </interface>
</definitions>
```

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

- A [local name] of `infault` or `outfault`

- A [namespace name] of "http://www.w3.org/2003/11/wsdl"

- One or more *attribute information items* amongst its [attributes] as follows:

  - A REQUIRED `name` *attribute information item* as described below in **2.5.2.1 name attribute information item with infault, or outfault [owner]** [p.26] .

  - An OPTIONAL `messageReference` *attribute information item* as described below in **2.5.2.2 messageReference attribute information item with infault, or outfault [owner]** [p.26] .

    If the {message exchange pattern} of the Interface Operation component has only one message with a given value for {direction}, the `messageReference` *attribute information item* is optional for the XML representation of any Fault Reference component with the same value for {direction} (if the *fault pattern* of the {message exchange pattern} is *fault-replaces-message*) or of any Fault Reference component with the opposite value for {direction} (if the *fault pattern* is *message-triggers-fault*).

  - An OPTIONAL `message` *attribute information item* as described below in **2.5.2.3 message attribute information item with infault, or outfault [owner]** [p.27] .
  - 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/2003/11/wsdl".

- Zero or more *element information item*s amongst its [children], in order, as follows:

  1. An OPTIONAL `documentation` *element information item* (see **5. Documentation** [p.55]).
  2. Zero or more namespace-qualified *element information item*s. The [namespace name] of such *element information item*s MUST NOT be "http://www.w3.org/2003/11/wsdl".

2.5.2.1 `name` *attribute information item* with `infault` , or `outfault` [owner]

The `name` *attribute information item* identifies the fault.

The `name` *attribute information item* has the following Infoset properties:

- A [local name] of `name`
- A [namespace name] which has no value

The type of the `name` *attribute information item* is *xs:NCName*.

2.5.2.2 `messageReference` *attribute information item* with `infault` , or `outfault` [owner]

The `messageReference` *attribute information item* identifies the message in the message exchange pattern of the given `operation` *element information item* to which this fault is related to.

The `messageReference` *attribute information item* has the following Infoset properties:

- A [local name] of `messageReference`
- A [namespace name] which has no value

The type of the `messageReference` *attribute information item* is *xs:NCName*.

2.5.2.3 `message` *attribute information item* with `infault` , or `outfault` [owner]

The `message` *attribute information item* refers, by QName, to an element declaration component.

The `message` *attribute information item* has the following Infoset properties:

- A [local name] of `message` .
- A [namespace name] which has no value.

The type of the `message` *attribute information item* is *xs:QName*.

2.5.3 Mapping Fault Reference's XML Representation to Component Properties

The mapping between the properties of the Fault Reference component (see **2.5.1 The Fault Reference Component** [p.24]) and the XML Representation of the message reference *element information item* (see **2.5.2 XML Representation of Fault Reference Component** [p.25]) is as described in Table 2-5 [p.27] .

Table 2-5. Mapping between Fault Reference Component Properties and XML Representation

Property	Mapping
{name}	The actual value of the `name` *attribute information item*
{message reference}	The actual value of the `messageReference` *attribute information item* if any; otherwise the {message reference} property of the message with the same {direction} from the {message exchange pattern} of the Interface Operation component, provided there is exactly one such message and the *fault pattern* of the {message exchange pattern} is *fault-replaces-message*; otherwise the {message reference} property of the message with the opposite {direction}, provided there is exactly one such message and the *fault pattern* is *message-triggers-fault*; otherwise empty.
{direction}	If the [local name] of the *element information item* is `infault` then "in", else if the [local name] of the *element information item* is `outfault` then "out".
{message}	The element declaration resolved to by the value of the `message` *attribute information item* if present, otherwise a similar construct in some type system as referred to by some other *attribute information item* if present, otherwise empty.

2.6 Feature

2.6.1 The Feature Component

A feature component describes an abstract piece of functionality typically associated with the exchange of messages between communicating parties. Although WSDL poses no constraints on the potential scope of such features, examples might include "reliability", "security", "correlation", and "routing". The presence of a feature component in a WSDL description indicates that the feature is either accepted or required in particular interactions.

Features in the Feature component are identified by their URI. Unless otherwise specified, recognizing a feature’s URI is assumed to be semantically equivalent to understanding the feature’s specification.

The properties of the Feature component are as follows:

- {name} A URI as defined by [*IETF RFC 2396 [p.57]*].
- {required} A boolean value.

2.6.2 XML Representation of Feature Component

```
<feature
      uri="xs:QName"
      required="xs:boolean"? >
  <documentation />?
</feature>
```

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

- A [local name] of `feature`
- A [namespace name] of "http://www.w3.org/2003/11/wsdl"
- One or more *attribute information item*s amongst its [attributes] as follows:
  - A REQUIRED `uri` *attribute information item* as described below in **2.6.2.1 uri attribute information item with feature [owner]** [p.28] .
  - An OPTIONAL `required` *attribute information item* as described below in **2.6.2.2 required attribute information item with feature [owner]** [p.29] .
  - 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/2003/11/wsdl".
- Zero or more *element information item*s amongst its [children], in order as follows:
  1. An OPTIONAL `documentation` *element information item* (see **5. Documentation** [p.55]).
  2. Zero or more namespace-qualified *element information item*s. The [namespace name] of such *element information item*s MUST NOT be "http://www.w3.org/2003/11/wsdl".

2.6.2.1 `uri` *attribute information item* with `feature` [owner]

The `uri` *attribute information item* specifies the URI of the feature.

The `uri` *attribute information item* has the following Infoset properties:

- A [local name] of `uri`
- A [namespace name] which has no value

The type of the `uri` *attribute information item* is `xs:anyURI` .

2.6.2.2 `required` *attribute information item* with `feature` [owner]

The `required` *attribute information item* specifies whether the use of the feature is mandatory or optional.

The `required` *attribute information item* has the following Infoset properties:

- A [local name] of `required`
- A [namespace name] which has no value

The type of the `required` *attribute information item* is `xs:boolean` .

2.6.3 Mapping Feature's XML Representation to Component Properties

The mapping between the properties of the Feature component (see **2.6.1 The Feature Component** [p.27]) and the XML Representation of the `feature` *element information item* (see **2.6.2 XML Representation of Feature Component** [p.28]) is as described in Table 2-6 [p.29] .

Table 2-6. Mapping between Feature Component Properties and XML Representation

Property	Mapping
{name}	The actual value of the `uri` *attribute information item*
{required}	If the value of the `required` *attribute information item* is "true" or "1", then "true", otherwise "false".

2.7 Property

2.7.1 The Property Component

A Property component describes the set of possible values for a particular property. The permissible values are specified by references to a Schema description. A property is typically used to control a feature's behavior. Properties, and hence property values, can be shared amongst features.

The properties of the Property component are as follows:

- {name} A URI as defined by [*IETF RFC 2396 [p.57]*].
- {required} A boolean value.
- {value constraint} A type definition constraining the value of the property.

2.7.2 XML Representation of Property Component

```
<property
      uri="xs:QName"
      required="xs:boolean"? >
  <documentation />?
  [ <value /> | <constraint /> ]
</property>
```

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

- A [local name] of `property`
- A [namespace name] of "http://www.w3.org/2003/11/wsdl"
- One or more *attribute information item*s amongst its [attributes] as follows:
    - A REQUIRED `uri` *attribute information item* as described below in **2.7.2.1 uri attribute information item with property [owner]** [p.31] .
    - An OPTIONAL `required` *attribute information item* as described below in **2.7.2.2 required attribute information item with feature [owner]** [p.31] .
    - 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/2003/11/wsdl".
- One or more *element information item*s amongst its [children], in order as follows:
    1. An OPTIONAL `documentation` *element information item* (see **5. Documentation** [p.55]).
    2. One REQUIRED *element information item* from among the following:
        - A `value` *element information item* as described in **2.7.2.3 value element information item with property [parent]** [p.31]
        - A `constraint` *element information item* as described in **2.7.2.4 constraint element information item with property [parent]** [p.31]
    3. 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/2003/11/wsdl".

2.7.2.1 `uri` *attribute information item* with `property` [owner]

The `uri` *attribute information item* specifies the URI of the property. It has the following Infoset properties:

- A [local name] of `uri`
- A [namespace name] which has no value

The type of the `uri` *attribute information item* is `xs:anyURI` .

2.7.2.2 `required` *attribute information item* with `feature` [owner]

The `required` *attribute information item* specifies whether the use of the property is mandatory or optional.

The `required` *attribute information item* has the following Infoset properties:

- A [local name] of `required`
- A [namespace name] which has no value

The type of the `required` *attribute information item* is `xs:boolean` .

2.7.2.3 `value` *element information item* with `property` [parent]

```
<property>
  <value>
    xs:anySimpleType
  </value>
</property>
```

The `value` *element information item* specifies the value of the property. It has the following Infoset properties:

- A [local name] of `value`
- A [namespace name] of "http://www.w3.org/2003/11/wsdl"

The type of the `value` *element information item* is `xs:anySimpleType` .

2.7.2.4 `constraint` *element information item* with `property` [parent]

```
<property>
  <constraint>
    xs:QName
  </constraint>
</property>
```

The `constraint` *element information item* specifies a constraint on the value of the property. It has the following Infoset properties:

- A [local name] of `constraint`
- A [namespace name] of "http://www.w3.org/2003/11/wsdl"

The type of the `constraint` *attribute information item* is `xs:QName` .

2.7.3 Mapping Property’s XML Representation to Component Properties

The mapping between the properties of the Property component (see **2.7.1 The Property Component** [p.29]) and the XML Representation of the `property` *element information item* (see **2.7.2 XML Representation of Property Component** [p.30]) is as described in Table 2-7 [p.32] .

Table 2-7. Mapping between Property Component Properties and XML Representation

Property	Mapping
{name}	The actual value of the `uri` *attribute information item*
{value constraint}	If the `constraint` *element information item* is present, the type referred to by the value of this *element information item*. Otherwise, if the `value` *element information item* is present, an anonymous simple type, whose base type is "xs:anySimpleType", with a single "enumeration" facet whose value is that of the *element information item*. Otherwise, "xs:anySimpleType".

2.8 Binding

2.8.1 The Binding Component

A Binding component describes a concrete message format and transmission protocol which may be used to define an endpoint (see **2.13 Endpoint** [p.45]). Binding components can be used to describe such information in a re-usable manner for any interface or specifically for a given interface. Furthermore, binding information MAY be specified on a per-operation basis (see **2.9.1 The Binding Operation Component** [p.35]) within an interface in addition to across all operations of an interface.

If a Binding component specifies operation-specific binding details (by including Binding Operation components) then it MUST specify an interface the Binding component applies to to indicate which interface the operations come from.

Conversely, a Binding component MAY omit any operation-specific binding details and MAY omit specifying an interface. Binding components that do not specify an interface MAY be used to specify operation-independent binding details for Service components with different interfaces.

No concrete binding details are given in this specification. The companion specification, *WSDL (Version 2.0): Bindings* [*WSDL 2.0 Bindings [p.58]*] defines such bindings for SOAP 1.2 [*SOAP 1.2 Part 1: Messaging Framework [p.59]*], HTTP [*IETF RFC 2616 [p.59]*] and MIME [*IETF RFC 2045 [p.58]*].

Other specifications MAY define additional binding details. Such specifications are expected to annotate the Binding component (and its sub-components) with additional properties and specify the mapping between those properties and the XML representation.

Bindings are named constructs and can be referred to by QName (see **2.16 QName resolution** [p.48]). For instance, Endpoint components refer to bindings in this way.

The properties of the Binding component are as follows:

- {name} An NCName as defined by [*XML Namespaces [p.58]*].
- {target namespace} A namespace name, as defined in [*XML Namespaces [p.58]*].
- {interface} An named interface definition indicating the interface for which binding information is being specified.
- {operations} A set of named binding operation definitions
- {features} A set of named feature definitions
- {properties} A set of named property definitions

For each Binding component in the {bindings} property of a definitions container, the combination of {name} and {target namespace} properties must be unique.

2.8.2 XML Representation of Binding Component

```
<definitions>
  <binding
        name="xs:NCName"
        interface="xs:QName"? >
    <documentation />?
    [ <feature /> | <property /> | <operation /> ]*
  </binding>
</definitions>
```

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

- A [local name] of `binding`
- A [namespace name] of "http://www.w3.org/2003/11/wsdl"
- One or more *attribute information item*s amongst its [attributes] as follows:
  - A REQUIRED `name` *attribute information item* as described below in **2.8.2.1 name attribute information item with binding [owner]** [p.34] .

    - An OPTIONAL `interface` *attribute information item* as described below in **2.8.2.2 interface attribute information item with binding [owner]** [p.34] .
    - 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/2003/11/wsdl".
- Zero or more *element information item*s amongst its [children], in order, as follows:
    1. An OPTIONAL `documentation` *element information item* (see **5. Documentation** [p.55]).
    2. Zero or more *element information item*s from among the following, in any order:
        - Zero or more `operation` *element information item*s (see **2.9.2 XML Representation of Binding Operation Component** [p.36]).
        - Zero or more `feature` *element information item*s (see **2.6.2 XML Representation of Feature Component** [p.28]).
        - Zero or more `property` *element information item*s (see **2.7.2 XML Representation of Property Component** [p.30]).
        - Zero or more namespace-qualified *element information item*s. The [namespace name] of such *element information item*s MUST NOT be "http://www.w3.org/2003/11/wsdl". Such *element information item*s are considered to be binding extension elements(see **2.8.2.3 Binding extension elements** [p.35]).

2.8.2.1 `name` *attribute information item* with `binding` [owner]

The `name` *attribute information item* together with the `targetNamespace` *attribute information item* of the `definitions` *element information item* forms the QName of the binding.

The `name` *attribute information item* has the following Infoset properties:

- A [local name] of `name`
- A [namespace name] which has no value

The type of the `name` *attribute information item* is *xs:NCName*.

2.8.2.2 `interface` *attribute information item* with `binding` [owner]

The `interface` *attribute information item* refers, by QName, to an Interface component.

The `interface` *attribute information item* has the following Infoset properties:

- A [local name] of `interface`

- A [namespace name] which has no value

The type of the `interface` *attribute information item* is *xs:QName*.

2.8.2.3 Binding extension elements

Binding extension elements are used to provide information specific to a particular binding. The semantics of such *element information item*s are defined by the specification for those *element information item*s. Such specifications are expected to annotate the Binding component with additional properties and specify the mapping between those properties and the XML representation.

2.8.3 Mapping Binding's XML Representation to Component Properties

The mapping between the properties of the Binding component (see **2.8.1 The Binding Component** [p.32]) and the XML Representation of the `binding` *element information item* (see **2.8.2 XML Representation of Binding Component** [p.33]) is as described in Table 2-8 [p.35] .

Table 2-8. Mapping between Binding Component Properties and XML Representation

Property	Mapping
{name}	The actual value of the `name` *attribute information item*
{target namespace}	The actual value of the `targetNamespace` *attribute information item* of the [parent] `definitions` *element information item*.
{interface}	The Interface component resolved to by the actual value of the `interface` *attribute information item*, if any.
{operations}	The set of Binding Operation components corresponding to the `operation` *element information item*s in [children], if any.
{features}	The set of Feature components corresponding to the `feature` *element information item*s in [children], if any.
{properties}	The set of Property components corresponding to the `property` *element information item*s in [children], if any.

2.9 Binding Operation

2.9.1 The Binding Operation Component

A Binding Operation component describes a concrete binding for a particular operation of an interface to a particular concrete message format. A particular operation of an interface is uniquely identified by the target namespace of the interface and the name of the operation within that interface.

The properties of the Binding Operation component are as follows:

- {name} An NCName as defined by [*XML Namespaces [p.58]*].
- {target namespace} A namespace name, as defined in [*XML Namespaces [p.58]*].
- {message references} A set of Binding Message Reference components
- {fault references} A set of Binding Fault Reference components

The tuple of properties {name} and {target namespace} identify the operation for which binding information is being specified.

For each Binding Operation component in the {operations} property of a Binding component the combination of {name} and {target namespace} properties must be unique. That is, one cannot define multiple bindings for the same operation.

2.9.2 XML Representation of Binding Operation Component

```
<definitions>
  <binding>
    <operation
          name="xs:QName" >
      <documentation />?
      [ <feature /> | <property /> |
        [ <input /> | <output /> | <infault /> | <outfault /> ]+
      ]*
    </operation>
  </binding>
</definitions>
```

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

- A [local name] of `operation`
- A [namespace name] of "http://www.w3.org/2003/11/wsdl"
- One or more *attribute information item*s amongst its [attributes] as follows:
    - A REQUIRED `name` *attribute information item* as described below in **2.9.2.1 name attribute information item with operation [owner]** [p.37] .
    - 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/2003/11/wsdl".
- Zero or more *element information item*s amongst its [children], in order, as follows:
    1. An OPTIONAL `documentation` *element information item* (see **5. Documentation** [p.55]).
    2. Zero or more *element information item*s from among the following, in any order:

- Zero or more `input` *element information item*s (see **2.10 Binding Message Reference** [p.38])
- Zero or more `output` *element information item*s (see **2.10 Binding Message Reference** [p.38])
- Zero or more `infault` *element information item*s (see **2.10 Binding Message Reference** [p.38])
- Zero or more `outfault` *element information item*s (see **2.10 Binding Message Reference** [p.38])
- Zero or more `feature` *element information item*s
- Zero or more `property` *element information item*s
- 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/2003/11/wsdl". Such *element information item*s are considered to be binding operation extension elements as described below (see **2.9.2.2 Binding Operation extension elements** [p.37]).

2.9.2.1 `name` *attribute information item* with `operation` [owner]

The `name` *attribute information item* identified a given `operation` *element information item* within a given `binding` *element information item*.

The `name` *attribute information item* has the following Infoset properties:

- A [local name] of `name`
- A [namespace name] which has no value

The type of the `name` *attribute information item* is *xs:QName*.

2.9.2.2 Binding Operation extension elements

Binding Operation extension elements are used to provide information specific to a particular operation in a binding. The semantics of such *element information item*s are defined by the specification for those *element information item*s. Such specifications are expected to annotate the Binding Operation component with additional properties and specify the mapping between those properties and the XML representation.

2.9.3 Mapping Binding Operation’s XML Representation to Component Properties

The mapping between the properties of the Binding Operation component (see **2.9.1 The Binding Operation Component** [p.35]) and the XML Representation of the `binding` *element information item* (see **2.9.2 XML Representation of Binding Operation Component** [p.36]) is as described in Table 2-9 [p.38] .

Table 2-9. Mapping between Binding Operation Component Properties and XML Representation

Property	Mapping
{name}	The value of the [local name] Infoset property of the qualified name which is the actual value of the `name` *attribute information item*.
{target namespace}	The value of the [namespace name] Infoset property of the qualified name which is the actual value of the `name` *attribute information item*.
{messages references}	The set of Binding Message Reference components corresponding to the `input` and `output` *element information item*s in [children], if any.
{fault references}	The set of Binding Fault Reference components corresponding to the `infault` and `outfault` *element information item*s in [children], if any.
{features}	The set of Feature components corresponding to the `feature` *element information item* in [children], if any.
{properties}	The set of Property components corresponding to the `property` *element information item* in [children], if any.

2.10 Binding Message Reference

2.10.1 The Binding Message Reference Component

A Binding Message Reference component describes a concrete binding for a particular message participating in an operation to a particular concrete message format.

The properties of the Binding Message Reference component are as follows:

- {message reference} An NCName as defined by [*XML Namespaces [p.58]*]. The value of this property identifies the message within the operation for which binding details are being specified.
- {direction} One of *in* or *out* indicating whether the message is coming to the service or going from the service, respectively.

For each Binding Message Reference component in the {message references} property of a Binding Operation component the {message reference} property must be unique. That is, the same message cannot be bound twice within the same operation.

2.10.2 XML Representation of Binding Message Reference Component

```
<definitions>
  <binding>
    <operation>
      <input
           messageReference="xs:NCName" >
        <documentation />?
      </input>
      <output
```

```
            messageReference="xs:NCName" >
        <documentation />?
      </output>
    </operation>
  </binding>
</definitions>
```

The XML representation for a Binding Message Reference component is an *element information item* with the following Infoset properties:

- A [local name] of `input` or `output` .
- A [namespace name] of "http://www.w3.org/2003/11/wsdl".
- One or more *attribute information item*s amongst its [attributes] as follows:
  - A REQUIRED `messageReference` *attribute information item* as described below in **2.10.2.1 messageReference attribute information item with input or output [owner]** [p.39] .
  - 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/2003/11/wsdl".
- Zero or more *element information item*s amongst its [children], in order, as follows:
  1. An OPTIONAL `documentation` *element information item* (see **5. Documentation** [p.55]).
  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/2003/11/wsdl". Such *element information item*s are considered to be binding message reference extension elements, as described below (see **2.10.2.2 Binding Message Reference extension elements** [p.39]).

2.10.2.1 `messageReference` *attribute information item* with `input` or `output` [owner]

The `messageReference` *attribute information item* has the following Infoset properties:

- A [local name] of `messageReference` .
- A [namespace name] which has no value.

The type of the `messageReference` *attribute information item* is *xs:NCName*.

2.10.2.2 Binding Message Reference extension elements

Binding Message Reference extension elements are used to provide information specific to a particular message in an operation. The semantics of such *element information item*s are defined by the specification for those *element information item*s. Such specifications are expected to annotate the Binding Message Reference component with additional properties and specify the mapping between those properties and the XML representation.

2.10.3 Mapping Binding Message Reference's XML Representation to Component Properties

The mapping between the properties of the Binding Message Reference component (see **2.10.1 The Binding Message Reference Component** [p.38]) and the XML Representation of the `binding` *element information item* (see **2.10.2 XML Representation of Binding Message Reference Component** [p.38]) is as described in Table 2-10 [p.40] .

Table 2-10. Mapping between Binding Message Reference Component Properties and XML Representation

Property	Mapping
{message reference}	The actual value of the `messageReference` *attribute information item*
{direction}	If the [local name] of the *element information item* is `input` then "in", else if the [local name] of the *element information item* is `output` then "out".

2.11 Binding Fault Reference

2.11.1 The Binding Fault Reference Component

A Binding Fault Reference component describes a concrete binding of a particular fault message of an operation to a particular concrete message format.

The properties of the Binding Fault Reference component are as follows:

- {name} An NCName as defined by [*XML Namespaces [p.58]*]. The value of this property MUST be the same as the {name} property of a Fault Reference component of the Interface Operation component for which binding details are being specified.
- {message reference} An NCName as defined by [*XML Namespaces [p.58]*]. The value of this property identifies the message within the operation for which this fault relates to and for which fault binding details are being specified.
- {direction} One of *in* or *out* indicating whether the message is coming to the service or going from the service, respectively.

For each Binding Fault Reference component in the {message references} property of a Binding Operation component the {message reference} property must be unique. That is, the same fault message cannot be bound twice within the same operation.

2.11.2 XML Representation of Binding Fault Reference Component

```
<definitions>
  <binding>
    <operation>
      <infault
            name="xs:NCName"
            messageReference="xs:NCName" >
        <documentation />?
      </infault>
      <outfault
            name="xs:NCName"
            messageReference="xs:NCName" >
        <documentation />?
      </outfault>
    </operation>
  </binding>
</definitions>
```

The XML representation for a Binding Fault Reference component is an *element information item* with the following Infoset properties:

- A [local name] of `infault` or `outfault` .
- A [namespace name] of "http://www.w3.org/2003/11/wsdl".
- Two or more *attribute information items* amongst its [attributes] as follows:
    - A REQUIRED `name` *attribute information item* as described below in **2.11.2.1 name attribute information item with infault or outfault [owner]** [p.42] .
    - A REQUIRED `messageReference` *attribute information item* as described below in **2.11.2.2 messageReference attribute information item with infault or outfault [owner]** [p.42] .
    - Zero or more namespace qualified *attribute information items*. The [namespace name] of such *attribute information items* MUST NOT be "http://www.w3.org/2003/11/wsdl".
- Zero or more *element information items* amongst its [children], in order, as follows:
    1. An OPTIONAL `documentation` *element information item* (see **5. Documentation** [p.55]).
    2. Zero or more namespace-qualified *element information items* amongst its [children]. The [namespace name] of such *element information items* MUST NOT be "http://www.w3.org/2003/11/wsdl". Such *element information items* are considered to be Binding Fault reference extension elements, as described below (see **2.11.2.3 Binding Fault Reference extension elements** [p.42]).

2.11.2.1 `name` *attribute information item* with `infault` or `outfault` [owner]

The `name` *attribute information item* has the following Infoset properties:

- A [local name] of `name` .
- A [namespace name] which has no value.

The type of the `name` *attribute information item* is *xs:NCName*.

2.11.2.2 `messageReference` *attribute information item* with `infault` or `outfault` [owner]

The `messageReference` *attribute information item* has the following Infoset properties:

- A [local name] of `messageReference` .
- A [namespace name] which has no value.

The type of the `messageReference` *attribute information item* is *xs:NCName*.

2.11.2.3 Binding Fault Reference extension elements

Binding Fault Reference extension elements are used to provide information specific to a particular message in an operation. The semantics of such *element information item*s are defined by the specification for those *element information item*s. Such specifications are expected to annotate the Binding Fault Reference component with additional properties and specify the mapping between those properties and the XML representation.

2.11.3 Mapping Binding Fault Reference’s XML Representation to Component Properties

The mapping between the properties of the Binding Fault Reference component (see **2.11.1 The Binding Fault Reference Component** [p.40]) and the XML Representation of the `binding` *element information item* (see **2.11.2 XML Representation of Binding Fault Reference Component** [p.41]) is as described in Table 2-11 [p.42] .

Table 2-11. Mapping between Binding Fault Reference Component Properties and XML Representation

Property	Mapping
{message reference}	The actual value of the `messageReference` *attribute information item*
{direction}	If the [local name] of the *element information item* is `infault` then "in", else if the [local name] of the *element information item* is `outfault` then "out".

2.12 Service

2.12.1 The Service Component

A Service component describes a set of endpoints (see **2.13 Endpoint** [p.45]) at which the single interface of the service is provided. The endpoints thus are in effect alternate places at which the service is provided.

Service are named constructs and can be referred to by QName (see **2.16 QName resolution** [p.48]).

The properties of the Service component are as follows:

- {name} An NCName as defined by [*XML Namespaces [p.58]*].
- {target namespace} A namespace name, as defined in [*XML Namespaces [p.58]*].
- {interface} An Interface component.
- {endpoints} A set of Endpoint components.

For each Service component in the {services} property of a definitions container, the combination of {name} and {target namespace} properties MUST be unique.

2.12.2 XML Representation of Service Component

```
<definitions>
  <service
        name="xs:NCName"
        interface="xs:QName"
    <documentation />?
    <endpoint />+
  </service>
</definitions>
```

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

- A [local name] of `service`
- A [namespace name] of "http://www.w3.org/2003/11/wsdl"
- Two or more *attribute information item*s amongst its [attributes] as follows:
  - A REQUIRED `name` *attribute information item* as described below in **2.12.2.1 name attribute information item with service [owner]** [p.44] .
  - A REQUIRED `interface` *attribute information item* as described below in **2.12.2.2 interface attribute information item with service [owner]** [p.44] .

- 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/2003/11/wsdl".

- One or more *element information item* amongst its [children], in order, as follows:

  1. An OPTIONAL `documentation` *element information item* (see **5. Documentation** [p.55]).

  2. One or more *element information item*s from among the following, in any order:

     - One or more `endpoint` *element information item*s (see **2.13.2 XML Representation of Endpoint Component** [p.45]

     - 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/2003/11/wsdl".

2.12.2.1 `name` *attribute information item* with `service` [owner]

The `name` *attribute information item* together with the `targetNamespace` *attribute information item* of the `definitions` *element information item* forms the QName of the service.

The `name` *attribute information item* has the following Infoset properties:

- A [local name] of `name`

- A [namespace name] which has no value

The type of the `name` *attribute information item* is *xs:NCName*.

2.12.2.2 `interface` *attribute information item* with `service` [owner]

The `interface` *attribute information item* identifies the interface that the service is an instantiation of.

The `interface` *attribute information item* has the following Infoset properties:

- A [local name] of `interface`

- A [namespace name] which has no value

The type of the `interface` *attribute information item* is *xs:QName.*.

2.12.3 Mapping Service’s XML Representation to Component Properties

The mapping between the properties of the Service component (see **2.12.1 The Service Component** [p.43]) and the XML Representation of the `service` *element information item* (see **2.12.2 XML Representation of Service Component** [p.43]) is as described in Table 2-12 [p.44] .

Table 2-12. Mapping between Service Component Properties and XML Representation

Property	**Mapping**
{name}	The actual value of the `name` *attribute information item*
{target namespace}	The actual value of the `targetNamespace` *attribute information item* of the [parent] `definitions` *element information item*
{interface}	The Interface component resolved to by the actual value of the `interface` *attribute information item*.
{endpoints}	The Endpoint components corresponding to the `endpoint` *element information item*s in [children] if any.

2.13 Endpoint

2.13.1 The Endpoint Component

An Endpoint component defines the particulars of a specific endpoint at which a given service is available.

Endpoint components are local to a given Service component; they cannot be referred to by QName.

The properties of the Endpoint component are as follows:

- {name} An NCName as defined by [*XML Namespaces [p.58]*].
- {binding} A named Binding component.

For each Endpoint component in the {endpoints} property of a Service component, the {binding} property (see **2.13.1 The Endpoint Component** [p.45]) MUST either be a Binding component with an unspecified {interface} property (see **2.8.1 The Binding Component** [p.32] or a Binding component with an {interface} property equal to the {interface} property of the Service component.

For each Endpoint component in the {endpoints} property of a Service component the {name} property must be unique.

2.13.2 XML Representation of Endpoint Component

```
<definitions>
  <service>
    <endpoint
          name="xs:NCName"
          binding="xs:QName" >
      <documentation />?
    </endpoint>
  </service>
</definitions>
```

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

- A [local name] of `endpoint` .
- A [namespace name] of "http://www.w3.org/2003/11/wsdl".
- Two or more *attribute information item*s amongst its [attributes] as follows:
    - A REQUIRED `name` *attribute information item* as described below in **2.13.2.1 name attribute information item with endpoint [owner]** [p.46] .
    - A REQUIRED `binding` *attribute information item* as described below in **2.13.2.2 binding attribute information item with endpoint [owner]** [p.46] .
    - 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/2003/11/wsdl".
- Zero or more *element information item* amongst its [children], in order, as follows:
    1. An OPTIONAL `documentation` *element information item* (see **5. Documentation** [p.55]).
    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/2003/11/wsdl". Such *element information item*s are considered to be endpoint extension elements (see **2.13.2.3 Endpoint extension elements** [p.47]).

2.13.2.1 `name` *attribute information item* with `endpoint` [owner]

The `name` *attribute information item* together with the `targetNamespace` *attribute information item* of the `definitions` *element information item* forms the QName of the endpoint.

The `name` *attribute information item* has the following Infoset properties:

- A [local name] of `name` .
- A [namespace name] which has no value.

The type of the `name` *attribute information item* is *xs:NCName*.

2.13.2.2 `binding` *attribute information item* with `endpoint` [owner]

The `binding` *attribute information item* refers, by QName, to a Binding component

The `binding` *attribute information item* has the following Infoset properties:

- A [local name] of `binding`

- A [namespace name] which has no value

The type of the `binding` *attribute information item* is *xs:QName*.

2.13.2.3 Endpoint extension elements

Endpoint extension elements are used to provide information specific to a particular endpoint in a server. The semantics of such *element information item*s are defined by the specification for those *element information item*s. Such specifications are expected to annotate the Endpoint component with additional properties and specify the mapping between those properties and the XML representation.

2.13.3 Mapping Endpoint's XML Representation to Component Properties

The mapping between the properties of the Endpoint component (see **2.13.1 The Endpoint Component** [p.45]) and the XML Representation of the `endpoint` *element information item* (see **2.13.2 XML Representation of Endpoint Component** [p.45]) is as described in Table 2-13 [p.47] .

Table 2-13. Mapping between Endpoint Component Properties and XML Representation

Property	Mapping
{name}	The actual value of the `name` *attribute information item*
{binding}	The Binding component resolved to by the actual value of the `binding` *attribute information item*.

2.14 Equivalence of Components

Two components of the same type are considered equivalent if the values of the properties of one component are the same as the values of the properties in the second component.

With respect to top-level components (Interfaces, Bindings and Services) this effectively translates to name-based equivalence given the constraints on names. That is, given two top-level components of the same type, if the {name} properties have the same value and the {target namespace} properties have the same values then the two components are in fact, the same component.

2.15 Symbol Spaces

This specification defines 3 symbol spaces, one for each top-level component type (Interface, Binding and Service).

Within a symbol space all names, that is the combination of {name} and {target namespace} properties, are unique. Between symbol spaces, the combination of these two properties need not be unique. Thus it is perfectly coherent to have, for example, a binding and an interface that have the same name.

When XML Schema is being used as one of the type systems for a WSDL description, then 6 other symbol spaces also exist, one for each of global element declarations, global attribute declarations, named model groups, named attribute groups, type definitions and key constraints, as defined by [*XML Schema:*

Structures [p.58]]. Other type systems may define additional symbol spaces.

2.16 QName resolution

In its serialized form WSDL makes significant use of references between components. Such references are made using the Qualified Name, or QName, of the component being referred to. QNames are a tuple, consisting of two parts; a namespace name and a local name. For example, in the case of an Interface component, the namespace name is represented by the {namespace name} property and the local name is represented by the {name} property.

QName references are resolved by looking in the appropriate property of the Definitions component. For example, to resolve a QName of an interface (as referred to by the `interface` *attribute information item* on a binding), the {interfaces} property of the Definitions component would be inspected.

If the appropriate property of the Definitions component does not contains a component with the required QName then the reference is a broken reference, it cannot be resolved.

3. Types

```
<definitions>
  <types>
    <documentation />?
    [extension elements]*
  </types>
</definitions>
```

At the abstract level, the {type definitions} and {element declarations} properties of **2.1.1 The Definitions Component** [p.8] are collections of imported and embedded schema components. By design, WSDL supports any schema language for which the syntax and semantics of import or embed have been defined. Support for the W3C XML Schema Description Language [*XML Schema: Structures [p.58]*],[*XML Schema: Datatypes [p.58]*] is required of all processors. Instances of WSDL may require support for an alternative schema language by using the standard `wsdl:required` *attribute information item* (any imported or embedded XML Schema *element information item*s may be regarded as having this *attribute information item* set). Only the XML Schema implementation is defined in this specification.

The schema components contained in the {type definitions} and {element declarations} properties of **2.1.1 The Definitions Component** [p.8] provide the type system used for Message Reference and Fault Reference components. Message Reference components indicate their structure and content by using the standard *attribute information item*s `message` , or for alternate schema languages in which these concepts do not map well, by using alternative *attribute information item* extensions. Fault Reference components behave similarly. Such extensions should define how they reference type system components. Such type system components MAY appear in additional collection properties on **2.1.1 The Definitions Component** [p.8] .

The `types` *element information item* encloses data type definitions used to define messages and has the following Infoset properties:

- A [local name] of `types` .
- A [namespace name] of "http://www.w3.org/2003/11/wsdl".
- Zero or more namespace qualified *attribute information item*s in its [attributes] property. The [namespace name] property of such *attribute information item*s MUST NOT be http://www.w3.org/2003/11/wsdl
- Zero or more *element information item*s amongst its [children] as follows:
    - An OPTIONAL `documentation` *element information item* (see **5. Documentation** [p.55]) in its [children] property.
    - Zero or more *element information item*s from among the following, in any order:
        - `xs:import` *element information item*s
        - `xs:schema` *element information item*s
        - Other namespace qualified *element information item*s whose namespace is NOT http://www.w3.org/2003/11/wsdl

3.1 Using W3C XML Schema Description Language

XML Schema MAY be used as the schema language, via import or embedding. Each method defines a different *element information item*. All processors MUST support XML Schema type definitions. Using the `xs:import` and/or `xs:schema` constructs is a necessary condition for making XML Schema components available to a WSDL description. That is, a WSDL description cannot refer to XML Schema components in a given namespace unless an `xs:import` and/or `xs:schema` statement for that namespace is present.

3.1.1 Importing XML Schema

Importing an XML Schema uses the `xs:import` mechanism defined by XML Schema [*XML Schema: Structures [p.58]*],[*XML Schema: Datatypes [p.58]*], with its syntax and semantics, plus some restrictions. The schema components defined in the imported schema are available for reference by QName (see **2.16 QName resolution** [p.48]). Note that only components defined in the schema itself and components included by it via `xs:include` are available to WSDL. Specifically components that the schema imports via `xs:import` are NOT available to WSDL.

A child *element information item* of the `types` *element information item* is defined with the Infoset properties as follows:

- A [local name] of "import".
- A [namespace name] of ""http://www.w3.org/2001/XMLSchema"".

- One or two *attribute information item*s as follows:
    - A REQUIRED `namespace` *attribute information item* as described below.
    - An OPTIONAL `schemaLocation` *attribute information item* as described below.

3.1.1.1 `namespace` *attribute information item*

The `namespace` *attribute information item* defines the namespace of the type definitions and element declarations imported from the referenced schema. The referenced schema MUST contain a `targetNamespace` *attribute information item* on its `xs:schema` *element information item* and the values of these two *attribute information item*s MUST be identical. It is an error to import a schema that does not have a `targetNamespace` *attribute information item* on its `xs:schema` *element information item*. Such schemas must first be included (using `xs:include`) in a schema that contains a `targetNamespace` *attribute information item* on its `xs:schema` *element information item*, which can then be either imported or inlined in the WSDL document. The `namespace` *attribute information item* has the following Infoset properties:

- A [local name] of namespace
- A [namespace name] which has no value.

The type of the `namespace` *attribute information item* is *xs:anyURI*.

3.1.1.2 `schemaLocation` *attribute information item*

The `schemaLocation` *attribute information item*, if present, provides a hint to the processor as to where the schema may be located. Caching and cataloging technologies may provide better information than this hint. The `schemaLocation` *attribute information item* has the following infoset properties:

- A [local name] of schemaLocation.
- A [namespace name] which has no value.

The type of the `schemaLocation` *attribute information item* is *xs:anyURI*.

3.1.2 Embedding XML Schema

Embedding an XML schema uses the existing top-level `xs:schema` *element information item*. It may be viewed as simply cutting and pasting an existing, stand-alone schema, to a location inside the types *element information item*. The schema components defined in the embedded schema are available to WSDL for reference by QName (see **2.16 QName resolution** [p.48]). Note that only components defined in the schema itself and components included by it via `xs:include` are available to WSDL. Specifically components that the schema imports via `xs:import` are NOT available to WSDL.

The `xs:schema` *element information item* has the following Infoset properties:

- A [local name] of schema.
- A [namespace name] of "http://www.w3.org/2001/XMLSchema".
- A REQUIRED `targetNamespace` *attribute information item*, amongst its [attributes] as described below.
- Additional *attribute information item*s as specified for the `xs:schema` *element information item* in the XML Schema specification.
- Child *element information item*s as specified for the `xs:schema` *element information item* in the XML Schema specification.

3.1.2.1 `targetNamespace` *attribute information item*

The `targetNamespace` *attribute information item* defines the namespace of the type definitions and element declarations embedded in this schema. WSDL modifies the XML Schema definition of the `xs:schema` *element information item* to make this *attribute information item* required. The `targetNamespace` *attribute information item* has the following infoset properties:

- A [local name] of targetNamespace.
- A [namespace name] which has no value.

The type of the `targetNamespace` *attribute information item* is *xs:anyURI*.

3.1.3 References to Element Declarations and Type Definitions

Whether embedded or imported, the element declarations present in a schema may be referenced from a Message Reference or Fault Reference component.

A named, global `xs:element` declaration may be referenced from the `message` *attribute information item* of an `input` or `output` *element information item*. The QName is constructed from the `targetNamespace` of the schema and the content of the `name` *attribute information item* of the `xs:element` *element information item*. A `message` *attribute information item* may not refer to an `xs:simpleType` or an `xs:complexType` *element information item*.

Similarly, a named, global `xs:simpleType` or `xs:complexType` definition may be referenced by QName. The QName is constructed from the `targetNamespace` of the schema and the content of the `name` *attribute information item* of the `xs:simpleType` or `xsd:complexType` *element information item*.

3.2 Using Other Schema Languages

Since it is unreasonable to expect that a single schema language can be used to describe all possible Message Reference and Fault Reference component content and their constraints, WSDL allows alternate schema languages to be specified via extensibility elements. An extensibility *element information item* MAY appear under the `types` *element information item* to identify the schema language employed, and to locate the schema instance defining the grammar for Message Reference and Fault Reference components. Depending upon the schema language used, an *element information item* MAY be defined to allow embedding, if and only if the schema language can be expressed in XML. A specification of extension syntax for an alternative schema language MUST include an *element information item* definition, intended to appear as a child of the `wsdl:types` *element information item*, which references, names, and locates the schema instance (an "import" *element information item*). The extension specification SHOULD, if necessary, define additional properties of **2.1.1 The Definitions Component** [p.8] to hold the components of the referenced type system. It is expected that additional extensibility attributes for Message Reference and Fault Reference components will also be defined, along with a mechanism for resolving the values of those attributes to a particular imported type system component.

See **D. Examples of Specifications of Extension Elements for Alternative Schema Language Support.** [p.63] for examples of using other schema languages. These example reuse the {element declarations} and {type definitions} properties of **2.1.1 The Definitions Component** [p.8] and the `type` and `element` *attribute information item*s of `wsdl:part` .

4. Modularizing WSDL descriptions

This specification provides two mechanisms, described in this section, for modularizing WSDL descriptions. These mechanisms help to make WSDL descriptions clearer by allowing separation of the various components of a description. Such separation could be performed according to the level of abstraction of a given set of components, or according to the namespace affiliation required of a given set of components or according to some other grouping such as application applicability.

Both mechanisms work at the level of WSDL components and NOT at the level of XML Information Sets or XML 1.0 serializations.

4.1 Including Descriptions

```
<definitions>
  <include
        location="xs:anyURI" >
    <documentation />?
  </include>
</definitions>
```

The WSDL `include` *element information item* allows for the separation of different components of a service definition, from the same target namespace, into independent WSDL documents which can be merged as needed.

The WSDL `include` *element information item* is modeled after the XML Schema `include` *element information item* (see [*XML Schema: Structures [p.58]*], section 4.2.3 "References to schema components in the same namespace"). Specifically it can be used to include components from WSDL descriptions that share a target namespace with the including description. Components in included descriptions are part of the component model of the including description. The included components can be referenced by QName. Note that because all WSDL descriptions have a target namespace, no-namespace includes (sometimes known as chameleon includes) never occur in WSDL.

The `include` *element information item* has:

- A [local name] of `include` .
- A [namespace name] of "http://www.w3.org/2003/11/wsdl".
- One or more *attribute information item*s amongst its [attributes] as follows:
  - A REQUIRED `location` *attribute information item* as described below in **4.1.1 location attribute information item with include [owner]** [p.53] .
  - 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/2003/11/wsdl".
- Zero or more *element information item* amongst its [children], as follows:
  - An optional `documentation` *element information item* (see **5. Documentation** [p.55]).
  - 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/2003/11/wsdl".

4.1.1 `location` *attribute information item* with `include` [owner]

The `location` *attribute information item* has the following Infoset properties:

- A [local name] of `location` .
- A [namespace name] which has no value.

A `location` *attribute information item* is of type `xs:anyURI` . Its actual value is the location of some information for the namespace identified by the `targetNamespace` *attribute information item* of the containing `definitions` *element information item.*

The actual value of the `targetNamespace` *attribute information item* of the included WSDL document MUST match the actual value of the `targetNamespace` *attribute information item* of the `definitions` *element information item* which is the [parent] of the `include` *element information item.*

4.2 Importing Descriptions

```
<definitions>
  <import
        namespace="xs:anyURI"
        location="xs:anyURI"? >
    <documentation />?
  </import>
</definitions>
```

The WSDL `import` *element information item*, like the `include` *element information item* (see **4.1 Including Descriptions** [p.52]) also allows for the separation of the different components of a WSDL description into independent descriptions, but in this case with different target namespaces, which can be imported as needed. This technique helps writing clearer WSDL descriptions, by separation the definitions according to their level of abstraction, and maximizes reusability.

The WSDL `import` *element information item* is modeled after the XML Schema `import` *element information item* (see [*XML Schema: Structures [p.58]*], section 4.2.3 "References to schema components across namespaces"). Specifically it can be used to import components from WSDL descriptions that do not share a target namespace with the importing document. Components in imported descriptions are part of the component model of the importing description. The imported components can be referenced by QName. Using the `import` construct is a necessary condition for making components from another namespace available to a WSDL description. That is, a WSDL description cannot refer to components in a namespace other that the target namespace unless an import statement for that namespace is present.

The `import` *element information item* has the following Infoset properties:

- A [local name] of `import` .
- A [namespace name] of "http://www.w3.org/2003/11/wsdl".
- Two or more *attribute information item*s amongst its [attributes] as follows:
    - A REQUIRED `namespace` *attribute information item* as described below in **4.2.1 namespace attribute information item** [p.54] .
    - An OPTIONAL `location` *attribute information item* as described below in **4.2.2 location attribute information item with import [owner]** [p.55] .
    - 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/2003/11/wsdl".
- Zero or more *element information item* amongst its [children], as follows:
    - An optional `documentation` *element information item* (see **5. Documentation** [p.55]).
    - 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/2003/11/wsdl".

4.2.1 `namespace` *attribute information item*

The `namespace` *attribute information item* has the following Infoset properties:

- A [local name] of `namespace` .

- A [namespace name] which has no value.

The `namespace` *attribute information item* is of type `xs:anyURI` . Its actual value indicates that the containing WSDL document MAY contain qualified references to WSDL definitions in that namespace (via one or more prefixes declared with namespace declarations in the normal way). This value MUST NOT match the actual value of the enclosing WSDL document `targetNamespace` *attribute information item*. If the import statement results in the import of a WSDL document then the actual value of the `namespace` *attribute information item* MUST be identical to the actual value of the imported WSDL document's `targetNamespace` *attribute information item*.

4.2.2 `location` *attribute information item* with `import` [owner]

The `location` *attribute information item* has the following Infoset properties:

- A [local name] of `location` .
- A [namespace name] which has no value.

The `location` *attribute information item* is of type `xs:anyURI` . Its actual value is the location of some information for the namespace identified by the `namespace` *attribute information item*.

The `location` *attribute information item* is optional to allow WSDL components to be constructed from information other than serialized XML 1.0 and to allow for WSDL processors that have *a priori* knowledge of certain namespaces.

5. Documentation

```
<documentation>
  [extension elements]*
</documentation>
```

WSDL uses the optional `documentation` *element information item* as a container for human readable and/or machine processable documentation. The content of the *element information item* is arbitrary *character information items* and *element information item*s ("mixed" in XML Schema[*XML Schema: Structures [p.58]*]). The `documentation` *element information item* is allowed inside any WSDL *element information item*.

The `documentation` *element information item* has:

- A [local name] of `documentation` .
- A [namespace name] of "http://www.w3.org/2003/11/wsdl".
- Zero or more *attribute information item*s in its [attributes] property.
- Zero or more child *element information item*s in its [children] property.

- Zero or more *character information item*s in its [children] property.

6. Language Extensibility

The schema for WSDL has a two-part extensibility model based on namespace-qualified elements and attributes.

6.1 Element based extensibility

WSDL allows extensions to be defined in terms of *element information item*s. Where indicated herein, WSDL allows namespace-qualified *element information item*s whose [namespace name] is NOT "http://www.w3.org/2003/11/wsdl" to appear as a child of specific *element information item*s whose namespace name IS "http://www.w3.org/2003/11/wsdl". Such *element information item*s can be used to annotate WSDL constructs such as interface, operation, etc.

It is expected that extensions will want to add to the existing properties of components in the component model. The specification for an extension *element information item* should include definitions of any such properties and the mapping between the XML representation of the extension and the properties in the component model.

The WSDL schema also defines a base type for use by extensibility elements. Example 6-1 [p.56] shows the type definition. The use of this type as a base type is optional. The element declarations which serve as the heads of the defined substitution groups are all of type "xs:anyType".

Extensibility elements are commonly used to specify some technology specific binding. They allow innovation in the area of network and message protocols without having to revise the base WSDL specification. WSDL recommends that specifications defining such protocols also define any necessary WSDL extensions used to describe those protocols or formats.

An extension element is said to be *processed* if a WSDL processor decides (through whatever means) that its parent (an *element information item* in the "http://www.w3.org/2003/11/wsdl" namespace) will be processed. Note that it is possible for WSDL processors to process only a subset of a given WSDL document. For instance, a tool may wish to focus on interfaces and operations only, and will consequently skip bindings.

Example 6-1. Base type for extensibility elements

```
<xs:complexType name=’ExtensibilityElement’ abstract=’true’ >
  <xs:attribute ref=’wsdl:required’ use=’optional’ />
</xs:complexType>
```

6.1.1 Mandatory extensions

Extension elements can be marked as mandatory by annotating them with a `wsdl:required` *attribute information item* (see **6.1.2 required attribute information item** [p.57]) with a value of "true". Mandatory extensions are those that MUST be processed correctly by the WSDL processor. If a mandatory extension element is processed, the WSDL processor MUST either agree to fully abide by all the rules and semantics signaled by the extension element's qualified name, or immediate cease processing (fault). In particular, if the WSDL processor does not recognize the qualified name of the extension element, it MUST fault. If the WSDL processor recognizes the qualified name, and determines that the extension in question is incompatible with any other aspect of the document (including other required extensions), it MUST fault.

6.1.2 `required` *attribute information item*

WSDL provides a global *attribute information item* with the following Infoset properties:

- A [local name] of `required` .
- A [namespace name] of "http://www.w3.org/2003/11/wsdl".
- A [specified] property with a value of "true".

The type of the `required` is *xs:boolean*.

6.2 Attribute-based extensibility

WSDL allows qualified *attribute information item*s whose [namespace name] is NOT "http://www.w3.org/2003/11/wsdl" to appear on any *element information item* whose namespace name IS "http://www.w3.org/2003/11/wsdl". Such *attribute information item*s can be used to annotate WSDL constructs such as interfaces, bindings, etc.

WSDL does not provide a mechanism for marking extension *attribute information item*s as mandatory.

7. References

7.1 Normative References

[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 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.

[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 Namespaces]
Namespaces in XML, T. Bray, D. Hollander, and A. Layman, Editors. World Wide Web Consortium, 14 January 1999. This version of the XML Information Set Recommendation is http://www.w3.org/TR/1999/REC-xml-names-19990114. The latest version of Namespaces in XML is available at http://www.w3.org/TR/REC-xml-names.

[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.

[RFC 3023]
IETF "RFC 3023: XML Media Types", M. Murata, S. St. Laurent, D. Kohn, July 1998. (See *http://www.ietf.org/rfc/rfc3023.txt*.)

[WSDL MediaType]
IETF Internet Draft "The ’application/wsdl+xml’ media type", @@@. (Work to be done once we have consensus on the media type).

[WSDL 2.0 Bindings]
Web Services Description (WSDL) Version 1.2: Bindings, J-J. Moreau, J. Schlimmer, Editors. World Wide Web Consortium, June 2003. This version of the "Web Services Description Version 1.2: Bindings" Specification is available at http://www.w3.org/TR/2003/WD-wsdl12-bindings-20030611. The latest version of "Web Services Description Version 1.2: Bindings" is available at http://www.w3.org/TR/wsdl12-bindings

[WSDL 2.0 Message Exchange Patterns]
Web Services Description (WSDL) Version 2.0: Message Exchange Patterns, M. Gudgin, A. Lewis, and J. Schlimmer, Editors. World Wide Web Consortium, November 2003. This version of the "Web Services Description Version 2.0: Message Exchange Patterns" Specification is available at http://www.w3.org/TR/2003/WD-wsdl20-patterns-20031110. The latest version of "Web Services Description Version 2.0: Message Exchange Patterns" is available at http://www.w3.org/TR/wsdl20-patterns.

[WSDL 2.0 RDF Mapping]
Web Services Description (WSDL) Version 2.0: RDF Mapping, to be written.

7.2 Informative References

[IETF RFC 2045]
Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, N. Freed, N. Borenstein, Authors. Internet Engineering Task Force, November 1996. Available at http://www.ietf.org/rfc/rfc2045.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.

[SOAP 1.1]
Simple Object Access Protocol (SOAP) 1.1, D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Frystyk Nielsen, S. Thatte, D. Winer, Editors. World Wide Web Consortium, 8 May 2000. This version of the Simple Object Access Protocol 1.1 Note is http://www.w3.org/TR/2000/NOTE-SOAP-20000508. The latest version of Simple Object Access Protocol 1.1 is available at http://www.w3.org/TR/SOAP.

[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/.

[XML Linking]
XML Linking Language (XLink) Version 1.0, S. DeRose, E. Maler, D. Orchard, Editors. World Wide Web Consortium, 27 June 2001. This version of the XML Linking Language 1.0 Recommendation is http://www.w3.org/TR/2001/REC-xlink-20010627. The latest version of XML Linking Language 1.0 is available at http://www.w3.org/TR/xlink.

[WSDL 1.1]
Web Services Description Language (WSDL) 1.1, E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, Authors. World Wide Web Consortium, 15 March 2002. This version of the Web Services Description Language 1.1 Note is http://www.w3.org/TR/2001/NOTE-wsdl-20010315. The latest version of Web Services Description Language 1.1 is available at http://www.w3.org/TR/wsdl.

[WSDL 2.0 Primer]
Web Services Description (WSDL) Version 2.0: Primer, K. Liu, D. Booth, Editors. World Wide Web Consortium, 10 November 2003. The editors' version of the Web Services Description Version 2.0: Primer document is available from http://www.w3.org/2002/ws/desc/.

[WSD Requirements]
Web Services Description Requirements, J. Schlimmer, Editor. World Wide Web Consortium, 28 October 2002. This version of the Web Services Description Requirements document is http://www.w3.org/TR/2002/WD-ws-desc-reqs-20021028. The latest version of Web Services Description Requirements is available at http://www.w3.org/TR/ws-desc-reqs.

[XPointer Framework]
XPointer Framework,Paul Grosso, Eve Maler, Jonathan Marsh, Norman Walsh, Editors. World Wide Web Consortium, 22 November 2002. This version of the XPointer Framework Proposed Recommendation is http://www.w3.org/TR/2003/REC-xptr-framework-20030325/ The latest version of XPointer Framework is available at http://www.w3.org/TR/xptr-framework/.

A. The "application/wsdl+xml" Media Type

Editorial note: JJM	20021107
This was lifted from the SOAP 1.2 specification, and needs to be edited to reflect WSDL's own requirements. For example, the WG has not reached consensus on whether to use "text/xml", "text/wsdl+xml" or "application/wsdl+xml".	

This appendix defines the "application/wsdl+xml" media type which can be used to describe WSDL 2.0 documents serialized as XML. It is referenced by the corresponding IANA registration document [*WSDL MediaType [p.58]*].

A.1 Registration

MIME media type name:

application

MIME subtype name:

wsdl+xml

Required parameters:

none

Optional parameters:
charset

This parameter has identical semantics to the charset parameter of the "application/xml" media type as specified in [*RFC 3023 [p.58]*].

Encoding considerations:

Identical to those of "application/xml" as described in [*RFC 3023 [p.58]*], section 3.2, as applied to the WSDL document infoset.

Security considerations:

See section **A.2 Security considerations** [p.61] .

Interoperability considerations:

There are no known interoperability issues.

Published specification:

This document and [*WSDL 2.0 Bindings [p.58]*].

Applications which use this media type:

No known applications currently use this media type.

Additional information:

File extension:

WSDL documents are not required or expected to be stored as files.

Fragment identifiers:

Either a syntax identical to that of "application/xml" as described in [*RFC 3023 [p.58]*], section 5 or the syntax defined in [*WSDL 2.0 RDF Mapping [p.58]*].

Base URI:

As specified in [*RFC 3023 [p.58]*], section 6.

Macintosh File Type code:

TEXT

Person and email address to contact for further information:

@@@ <@@@@>

Intended usage:

COMMON

Author/Change controller:

The WSDL 2.0 specification set is a work product of the World Wide Web Consortium’s Web Service Description Working Group. The W3C has change control over these specifications.

A.2 Security considerations

Editorial note: JJM	20021107
Are there any security considerations other than the standard ones.	

This media type uses the "+xml" convention, it shares the same security considerations as described in [*RFC 3023 [p.58]*], section 10.

B. 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): Adi Sakala (IONA Technologies), Alan Davies (SeeBeyond), Allen Brookes (Rogue Wave Softwave), Amelia Lewis (TIBCO/Extensibility), Arthur Ryman (IBM), Bijan Parsia (University of Maryland), Dale Moberg (Cyclone Commerce), Dan Kulp (IONA Technologies), David Booth (W3C), Dietmar Gaertner (Software AG), Don Mullen (TIBCO Software), Erik Ackerman (Lexmark), Glen Daniels (Macromedia), Igor Sedukhin (Computer Associates), Ingo Melzer (DaimlerChrysler Research and Technology), Jacek Kopecky (Systinet), Jean-Jacques Moreau (Canon), Jeff Mischkinsky (Oracle Corporation), Jeffrey Schlimmer (Microsoft Corporation), Jerry Thrasher (Lexmark), Jim Hendler (University of Maryland), Jonathan Marsh (*Chair*, Microsoft Corporation), Kevin Canyang Liu (SAP), Lily Liu (webMethods, Inc.), Martin Gudgin (Microsoft Corporation), Michael Champion (Software AG), Michael Mahan (Nokia), Michael Mealling (Verisign), Mike Ballantyne (Electronic Data Systems), Mike Davoren (W. W. Grainger), Mike McHugh (W. W. Grainger), Paul Downey (BT), Philippe Le Hégaret (W3C), Prasad Yendluri (webMethods, Inc.), Roberto Chinnici (Sun Microsystems), Sandeep Kumar (Cisco Systems), Sanjiva Weerawarana (IBM), Steve Graham (Global Grid Forum), Steve Tuecke (Global Grid Forum), Tom Jordahl (Macromedia), Umit Yalcinalp (Oracle Corporation), Waqar Sadiq (Electronic Data Systems), William Vambenepe (Hewlett-Packard Company), Youenn Fablet (Canon)

Previous members were: Aaron Skonnard (DevelopMentor), Jochen Ruetschlin (DaimlerChrysler Research and Technology), Keith Ballinger (Microsoft), Krishna Sankar (Cisco Systems), Mario Jeckle (DaimlerChrysler Research and Technology), Pallavi Malu (Intel Corporation), 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'Échangeur), Johan Pauhlsson (L'Échangeur), Mark Jones (AT&T), Steve Lind (AT&T), Sandra Swearingen (U.S. Department of Defense, U.S. Air Force)

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

C. Migrating from WSDL 1.1 to WSDL 2.0 (Non-Normative)

This section will attempt to document some of the migration concerns of going from WSDL 1.1 to WSDL 2.0. We do not claim that all migration problems will be addressed here.

C.1 Operation Overloading

WSDL 1.1 supported operation overloading and WSDL 2.0 removes it. This section will provide some rationale for it and provide hints on how to work around some scenarios.

C.2 PortTypes

Port types have been renamed to interfaces. We now have interface inheritance.

C.3 Ports

Ports have been renamed to endpoints.

D. Examples of Specifications of Extension Elements for Alternative Schema Language Support. (Non-Normative)

Editorial note: RRC	20030915
This section hasn't yet been updated to reflect the removal of the message construct from the language.	

D.1 DTD

A DTD may be used as the schema language for WSDL. It may not be embedded; it must be imported. A namespace must be assigned. DTD types appear in the {element declarations} property of **2.1.1 The Definitions Component** [p.8] and may be referenced from `wsdl:part` only by the `element` *attribute information item*; DTDs do not have a construct corresponding to the contents of the {type definitions} property of the **2.1.1 The Definitions Component** [p.8] .

The prefix, dtd, used throughout the following is mapped to the namespace URI "http://www.example.org/dtd/".

The `dtd:import` *element information item* references an external Document Type Definition, and has the following infoset properties:

- A [local name] of import.
- A [namespace name] of "http://www.example.org/dtd".
- One or two *attribute information items*, as follows:
  - A REQUIRED `namespace` *attribute information item* as described below.
  - An OPTIONAL `location` *attribute information item* as described below.

D.1.1 `namespace` *attribute information item*

The `namespace` *attribute information item* sets the namespace to be used with all imported element definitions described in the DTD. It has the following infoset properties:

- A [local name] of namespace.
- A [namespace name] which has no value.

The type of the `namespace` *attribute information item* is *xs:anyURI*.

The WSDL author should ensure that a prefix is associated with the namespace at the proper scope (probably document scope).

D.1.2 `location` *attribute information item*

The `location` *attribute information item*, if present, provides a hint to the processor as to where the DTD may be located. Caching and cataloging technologies may provide better information than this hint. The `location` *attribute information item* has the following infoset properties:

- A [local name] of location.
- A [namespace name] which has no value.

The type of the `location` *attribute information item* is *xs:anyURI*.

D.1.3 References to Element Definitions

The `message` *attribute information item* MUST be used when referring to an element definition (<!ELEMENT>) from a Message Reference; referring to an element definition from a Fault Reference component is similar. The value of the element definition MUST correspond to the content of the `namespace` *attribute information item* of the `dtd:import` *element information item*. The local name part must correspond to an element defined in the DTD.

Note that this pattern does not attempt to make DTDs namespace-aware. It applies namespaces externally, in the import phase.

D.2 RELAX NG

A RELAX NG schema may be used as the schema language for WSDL. It may be embedded or imported; import is preferred. A namespace must be specified; if an imported schema specifies one, then the [actual value] of the `namespace` *attribute information item* in the `import` *element information item* must match the specified namespace. RELAX NG provides both type and element definitions which appear in the {type definitions} and {element declarations} properties of **2.1.1 The Definitions Component** [p.8] respectively. The following discussion supplies the prefix rng which is mapped to the URI "http://www.relaxng.org/ns/structure/1.0".

D.2.1 Importing RELAX NG

Importing a RELAX NG schema uses the rng:include mechanism defined by RNG, with restrictions on its syntax and semantics. A child *element information item* of the `types` *element information item* is defined with the Infoset properties as follows:

- A [local name] of include.
- A [namespace name] of "http://www.relaxng.org/ns/structure/1.0".
- Two *attribute information item*s as follows:
  - A REQUIRED `ns` *attribute information item* as described below.
  - An OPTIONAL `href` *attribute information item* as described below.
  - Additional *attribute information item*s as defined by the RNG specification.

Note that WSDL restricts the `rng:include` *element information item* to be empty. That is, it cannot redefine `rng:start` and `rng:define` *element information items*; it may be used solely to import a schema.

D.2.1.1 `ns` *attribute information item*

The `ns` *attribute information item* defines the namespace of the type and element definitions imported from the referenced schema. If the referenced schema contains an `ns` *attribute information item* on its `grammar` *element information item*, then the values of these two *attribute information item*s must be identical. If the imported grammar does not have an `ns` *attribute information item* then the namespace specified here is applied to all components of the schema as if it did contain such an *attribute information item*. The `ns` *attribute information item* contains the following Infoset properties:

- A [local name] of ns.
- A [namespace name] which has no value.

The type of the `ns` *attribute information item* is *xs:anyURI*.

D.2.1.2 `href` *attribute information item*

The `href` *attribute information item* must be present, according to the rules of the RNG specification. However, WSDL allows it to be empty, and considers it only a hint. Caching and cataloging technologies may provide better information that this hint. The `href` *attribute information item* has the following Infoset properties:

- A [local name] of href.
- A [namespace name] which has no value.

The type of the `href` *attribute information item* is *xs:anyURI*.

D.2.2 Embedding RELAX NG

Embedding an RNG schema uses the existing top-level `rng:grammar` *element information item*. It may be viewed as simply cutting and pasting an existing, stand-alone schema to a location inside the `wsdl:types` *element information item*. The `rng:grammar` *element information item* has the following Infoset properties:

- A [local name] of grammar.
- A [namespace name] of "http://www.relaxng.org/ns/structure/1.0".
- A REQUIRED `ns` *attribute information item*s as described below.
- Additional *attribute information item*s as specified for the `rng:grammar` *element information item* in the RNG specification.
- Child *element information item*s as specified for the `rng:grammar` *element information item* in the RNG specification.

D.2.2.1 `ns` *attribute information item*

The `ns` *attribute information item* defines the namespace of the type and element definitions embedded in this schema. WSDL modifies the RNG definition of the `rng:grammar` *element information item* to make this *attribute information item* required. The `ns` *attribute information item* has the following infoset properties:

- A [local name] of ns.
- A [namespace name] which has no value.

The type of the `ns` *attribute information item* is *xs:anyURI*.

D.2.3 References to Element Declarations

Whether embedded or imported, the element definitions present in a schema may be referenced from a Message Reference or Fault Reference component.

A named rng:define definition MUST NOT be referenced from the Message Reference or Fault Reference components.

A named Relax NG element declaration MAY be referenced from a Message Reference or Fault Reference component. The QName is constructed from the namespace (`ns` *attribute information item*) of the schema and the content of the `name` *attribute information item* of the `element` *element information item* An `element` *attribute information item* MUST NOT be used to refer to an `rng:define` *element information item*.

E. Part 1 Change Log (Non-Normative)

E.1 WSDL Specification Changes

Date	Author	Description
20031105	JS	Added point to attributes task force recommendation accepted by the working group.
20031104	JS	Mapping to component model for {message} of Fault Reference component indicated that `message` *attribute information item* was optional, but the pseudo syntax and XML representation indicated it was required. Made uniformly optional to allow other type systems as was previously done for {message} of Message Reference component.
20031104	JS	Renamed interface /operation /{input,output} /@body to ./@message and interface /operation /{infault,outfault} /@details to ./@message per 4 Nov face-to-face decision.
20031104	JS	Made interface /operation /{input,output,infault,outfault} /@messageReference optional per 4 Nov face-to-face decision.
20031104	JS	Removed interface/operation/{input,output}/@header per 4 Nov face-to-face decision.
20031102	SW	Updated fault reference components to indicate that if operation’s MEP uses MTF then the fault is in the opposite direction as the referenced message and if it use FRM then its in the same direction. Per 10/30 telecon decision.
20031102	SW	Updated operation styles terminology per message #57 of Oct. and the RPC style rules per message #58 of Oct. per decision on 10/30 telecon to consider those status quo.
20031102	SW	Clarified wording in operation styles discussion to better explain the use of the {style} attribute.
20031102	SW	Clarified wording in XML <-> component model mapping section for message reference components to say that {body} and {headers} may not have a value.
20031102	SW	Made interface/operation/(input|output)/@messageReference REQUIRED per 10/30 telecon decision.
20031028	SW	Renamed to wsdl20.xml and updated contents.
20031028	SW	Updated bindings.
20031025	SW	Updated faults.
20031013	JJM	Moved appendix C to a separate document, as per 24 Sep 2003 meeting in Palo Alto, CA.

20031003	SW	Softened <documentation> wording to allow machine processable documentation.
20031002	SW	Changed binding/operation/@name to QName per edtodo.
20030930	SW	Added placeholders for set-attr/get-attr operation styles.
20030929	SW	Inserted Glen Daniels' feature text.
20030919	RRC	Removed import facility for chameleon schemas and added a description of a workaround.
20030918	JJM	Changed message pattern to message exchange pattern, as per WG resolution on 18 Sep. 2003
20030916	RRC	Added editorial note for the missing RPC encoding style.
20030915	RRC	Yet more updates for REQUIRED, OPTIONAL; updated section 3 to reflect the removal of "wsdl:message".
20030911	RRC	More updates for REQUIRED, OPTIONAL; removed diff markup; fixed example C.4.
20030911	RRC	Renamed message reference "name" attribute and property to "messageReference"; fixed incorrect reference to "fault" element in the binding operation section.
20030910	SW	Fixed message references and added proper use of REQUIRED etc. for the part I've gone through so far.
20030910	SW	Updating spec; fixed up interface operation component more.
20030808	JCS	Fixed errors found by IBM\Arthur.
20030804	JCS	Removed Message component per 30 July-1 Aug meeting.
20030803	JCS	Replaced substitution groups with xs:any namespace='##other' per 3 July, 17 July, and 24 July telecons.
20030801	JCS	Made binding/@interface optional per 31 July meeting.
20030724	JCS	Remove @targetResource per 17 July 2003 telecon.
20030612	JJM	Incorporate revised targetResource definition, as per 12 June 2003 telcon.
20030606	JJM	Refer to the two graphics by ID. Indicate pseudo-schemas are not normative.
20030604	JJM	Fixed figures so they don't appear as tables. Fixed markup so it validates.
20030603	JCS	Plugged in jmarsh auto-generated schema outlines
20030529	MJG	Fixed various issues with the XmlRep portions of the spec
20030527	MJG	Added text to **2.2.1 The Interface Component** [p.11] and **2.2.3 Mapping Interface's XML Representation to Component Properties** [p.14] indicating that recursive interface extension is not allowed.

20030523	JJM	Added pseudo-syntax to all but Type and Modularizing sections.
20030523	JJM	Added the "interface" and "targetResource" attribute on <service>.
20030523	JJM	Fixed miscellaneous typos (semi-colon instead of colon, space after parenthesis, etc.).
20030523	JJM	Rewrote the service-resource text and merge it with the introduction.
20030522	JCS	s/set of parts/list of parts/.
20030514	JJM	Updated the service-resource figure, and split the diagram into two.
20030512	JJM	Added service-resource drawing and description.
20030512	JJM	Added syntax summary for the Interface component.
20030428	MJG	Various edits to **3. Types** [p.48] , **D. Examples of Specifications of Extension Elements for Alternative Schema Language Support.** [p.63] to accommodate other type systems and spell out how extensibility elements/attributes play out in such scenarios.
20030428	MJG	Added text to **1.2 Notational Conventions** [p.6] regarding normative nature of schema and validity of WSDL documents
20030411	JJM	Allowed features and properties at the interface, interface operation, binding and binding operation levels, as agreed at the Boston f2f http://lists.w3.org/Archives/Public/www-ws-desc/2003Mar/0019.html.
20030411	JJM	Incorporate features and properties’ text from separate document and merged change logs
20030313	MJG	Changed title to include ’part 1’
20030313	MJG	Changed port to endpoint
20030313	MJG	Changed type to interface in binding
20030313	MJG	Changed mep to pattern and message exchange pattern to message pattern
20030313	MJG	Added text to **C.2 PortTypes** [p.63]
20030313	MJG	Changed portType to interface
20030407	JJM	Refined and corrected the definitions for features and properties.
20030304	JJM	Filled in blank description of Feature and Property component.
20030303	MJG	Skeleton Feature and Property components

20030305	MJG	Merged ComponentModelForMEPs branch (1.46.2.5) into main branch (1.54). Below is change log from the branch:

Date	Author	Description
20030220	MJG	Minor wording change at suggestion of JJM
20030212	MJG	Updated component model to include Fault Reference component. Associated changes to Port Type Operation component
20030211	MJG	Changes to component model to support MEPs

20030228	MJG	Updated **4.2 Importing Descriptions** [p.53] to be consistent in layout with other XML rep sections. Detailed that documentation and extensibility attributes are allowed, per schema
20030228	MJG	Updated **4.1 Including Descriptions** [p.52] to be consistent in layout with other XML rep sections. Detailed that documentation and extensibility attributes are allowed, per schema
20030228	MJG	Updated **2.8.2 XML Representation of Binding Component** [p.33] to list type attribute
20030217	MJG	Minor edits to wording in **2.3.1 The Interface Operation Component** [p.15]
20030213	MJG	Added xlink nsdecl to spec element
20030213	MJG	Incorporated text from dbooths proposal on semantics, per decision 20021031
20030213	MJG	Merged operationnames branch (1.37.2.3) into main branch (1.46). Below is the change log from the branch.

Date	Author	Description
20030130	MJG	Updated binding section to match changes to port type section WRT operation names
20030130	MJG	Added best practice note on operation names and target namespaces to **2.3.1 The Interface Operation Component** [p.15]
20030122	MJG	Started work on making operations have unique names

20030213	MJG	Change name of {message exchange pattern} back to {variety} to consolidate changes due to MEP proposal
20030206	MJG	Updated Appendix A to refer to Appendix C
20030204	MJG	Tidied up appendix C

20030203	MJG	Incorporated resolution to R120
20030124	MJG	Fixed error in **2.4.2 XML Representation of Message Reference Component** [p.22] which had name *attribute information item* on input, output and fault *element information item* being mandatory. Made it optional.
20030123	JJM	Change name of {variety} property to {message exchange pattern}
20030130	MJG	Updated binding section to match changes to port type section WRT operation names
20030130	MJG	Added best practice note on operation names and target namespaces to **2.3.1 The Interface Operation Component** [p.15]
20030122	MJG	Started work on making operations have unique names
20030122	MJG	Added some <emph>, <el>, <att>, &AII;, &EII;, <el> markup
20030120	MJG	Incorporated Relax NG section from Amy’s types proposal
20030120	MJG	Incorporated DTD section from Amy’s types proposal
2003020	MJG	Incorporated Amy’s types proposal except annexes
20030118	MJG	Made some changes related to extensibility
20030118	MJG	Amended content model for operation to disallow fault element children in the input-only and output-only cases
20030118	MJG	Removed {extension} properties from Binding components and Port components. Added text relating to how extension elements are expected to annotate the component model.
20030117	MJG	Made further edits related to extensibility model now using substitution groups
20030117	MJG	Added initial draft of section on QName resolution
20030117	MJG	Reworked section on extensibility
20030116	MJG	Added text regarding multiple operations with the same {name} in a single port type
20030116	MJG	Added section on symbol spaces
20030116	MJG	Removed various ednotes
20030116	MJG	Added section on component equivalence
20030116	MJG	More work on include and import
20021201	MJG	Did some work on wsdl:include
20021127	MJG	Added placeholder for wsdl:include

20021127	MJG	Cleaned up language concerning `targetNamespace` *attribute information item* **2.1.2.1 targetNamespace attribute information item** [p.10]
20021127	MJG	changed the language regarding extensibility elements in **2.1.2 XML Representation of Definitions Component** [p.9] .
20021127	MJG	Moved all issues into issues document (../issues/wsd-issues.xml)
20021127	MJG	Removed name attribute from definitions element
20021127	MJG	Removed ’pseudo-schema’
20021121	JJM	Updated media type draft appendix ednote to match minutes.
20021111	SW	Added appendix to record migration issues.
20021107	JJM	Incorporated and started adapting SOAP’s media type draft appendix.
20021010	MJG	Added port type extensions, removed service type.
20020910	MJG	Removed parameterOrder from spec, as decided at September 2002 FTF
20020908	MJG	Updated parameterOrder description, fixed some spelling errors and other types. Added ednote to discussion of message parts
20020715	MJG	AM Rewrite
20020627	JJM	Changed a few remaining <emph> to either <att> or <el>, depending on context.
20020627	SW	Converted portType stuff to be infoset based and improved doc structure more.
20020627	SW	Converted message stuff to be infoset based and improved doc structure more.
20020625	SW	Mods to take into account JJM comments.
20020624	JJM	Fixed spec so markup validates.
20020624	JJM	Upgraded the stylesheet and DTD
20020624	JJM	Added sections for references and change log.
20020624	JJM	Removed Jeffrey from authors :-(Added Gudge :-)
20020620	SW	Started adding abstract model
20020406	SW	Created document from WSDL 1.1