Web Services Description Language (WSDL) Version 1.2

W3C Working Draft 3 March 2003

This version:
http://www.w3.org/TR/2003/WD-wsdl12-20030303
Latest version:
http://www.w3.org/TR/wsdl12
Previous versions:
http://www.w3.org/TR/2003/WD-wsdl12-20030124
Editors:
Roberto Chinnici, Sun Microsystems
Martin Gudgin, Microsoft
Jean-Jacques Moreau, Canon
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 1.2, 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. The latest status of this document series is maintained at the W3C.

This is a W3C Working Draft of the Web Services Description Language (WSDL) 1.2 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.

This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to use W3C Working Drafts as reference material or to cite them as other than "work in progress". This is work in progress and does not imply endorsement by, or the consensus of, either W3C

or members of the Web Services Description Working Group.

Comments on this document are invited and are to be sent to public-ws-desc-comments@w3.org (public archive). It is inappropriate to send discussion emails to this address. Discussion of this document takes place on 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.

A list of current W3C Recommendations and other technical documents can be found at http://www.w3.org/TR.

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 companion specification, *WSDL Version 1.2: Bindings* [*WSDL 1.2 Bindings [p.46]*] defines a language for describing such concrete details for SOAP 1.2 [*SOAP 1.2 Part 1: Messaging Framework [p.47]*], HTTP [*IETF RFC 2616 [p.47]*] and MIME [*IETF RFC 2045 [p.47]*].

WSDL describes Web services starting with the *messages* that are exchanged between the service provider and requestor. The messages themselves are described abstractly and then bound to a concrete network protocol and message format. A message consists of a collection of typed data items. An exchange of messages between the service provider and requestor are described as an *operation*. A collection of operations is called a *port type*. A *service* contains a collection of *ports*, where each port is an implementation of a portType, which includes all the concrete details needed to interact with the service.

1.1 Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [*IETF RFC 2119 [p.46]*].

This specification uses properties from the XML Information Set [*XML Information Set [p.46]*]. Such properties are denoted by square brackets, e.g. [namespace name].

This specification uses namespace prefixes throughout; they are listed in **Table 1** [p.6] . Note that the choice of any namespace prefix is arbitrary and not semantically significant (see [*XML Information Set [p.46]*]).

Table 1: Prefixes and Namespaces used in this specification.

Prefix	Namespace	Notes
wsdl	"http://www.w3.org/2003/03/wsdl"	A normative XML Schema [*XML Schema: Structures [p.46]*], [*XML Schema: Datatypes [p.46]*] document for the "http://www.w3.org/2003/03/wsdl" namespace can be found at http://www.w3.org/2003/03/wsdl.
soap12	"http://www.w3.org/2003/01/wsdl/soap12"	Defined by WSDL 1.2: Bindings [*WSDL 1.2 Bindings [p.46]*].
http	"http://www.w3.org/2003/01/wsdl/http"	
mime	"http://www.w3.org/2003/01/wsdl/mime"	
xs	"http://www.w3.org/2001/XMLSchema"	Defined in the W3C XML Schema specification [*XML Schema: Structures [p.46]*], [*XML Schema: Datatypes [p.46]*].
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.46]*].

All parts of this specification are normative, with the exception of examples and sections explicitly marked as "Non-Normative".

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 messages, port types, 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:

- {messages} A set of named message definitions
- {port types} A set of named port type 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

2.1.2 XML Representation of Definitions Component

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.37] 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/2003/03/wsdl".
- One or more *attribute information item*s amongst its [attributes] as follows;
    - A `targetNamespace` *attribute information item* as described below in **2.1.2.1 targetNamespace attribute information item** [p.8] .

    - 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/2003/03/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.38]).
    2. Zero or more *element information item*s from among the following, in any order:
        1. Zero or more `include` *element information item*s (see **4.1 Including Descriptions** [p.36])
        2. Zero or more `import` *element information item*s (see **4.2 Importing Descriptions** [p.37])
    3. An optional `types` *element information item* (see **3. Types** [p.33]).
    4. Zero or more *element information item*s from among the following, in any order:
        1. `message` *element information item*s (see **2.2.2 XML Representation of Message Component** [p.10]).
        2. `portType` *element information item*s (see **2.4.2 XML Representation of Port Type Component** [p.14]).
        3. `binding` *element information item*s (see **2.7.2 XML Representation of Binding Component** [p.22]).
        4. `service` *element information item*s (see **2.10.2 XML Representation of Service Component** [p.28]).
- Zero or more namespace qualified *element information item*s amongst its [children]. Such *element information item*s MUST be a member of one of the element substitution groups allowed at the top-level of a WSDL document as described in **6. Language Extensibility** [p.38] .

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*. Messages, port types, 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.7]) and the XML Representation of the `definitions` *element information item* (see **2.1.2 XML Representation of Definitions Component** [p.7]) is described in **Table 2** [p.9] .

Table 2: Mapping between Definitions Component Properties and XML Representation

Property	Mapping
{messages}	The message definitions corresponding to all the `message` *element information item*s in the [children] of the `definitions` *element information item*, if any, plus any included or imported definitions (see **4. Modularizing WSDL descriptions** [p.36]).
{port types}	The port type definitions corresponding to all the `portType` *element information item*s in the [children] of the `definitions` *element information item*, if any, plus any included or imported definitions (see **4. Modularizing WSDL descriptions** [p.36]).
{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 definitions (see **4. Modularizing WSDL descriptions** [p.36]).
{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 definitions (see **4. Modularizing WSDL descriptions** [p.36]).
{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 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 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 Message

2.2.1 The Message Component

A message component describes the abstract format of a particular message that a Web service sends or receives. The format of a message is typically described in terms of XML *element information item*s and *attribute information item*s. A message binding (see **2.7 Binding** [p.21]) describes how the abstract content is mapped into a concrete format. However, in some cases, the abstract definition may match the

concrete representation very closely or exactly for one or more bindings. Such bindings will supply little or no mapping information. However, another binding of the same message definition may require extensive mapping information. For this reason, it is not until the binding is inspected that one can determine "how abstract" a message really is.

Messages are named constructs and can be referred to by QName (see **2.14 QName resolution** [p.32]). For instance, port type components refer to messages in this way (see **2.6 Message Reference** [p.19]).

The properties of the Message Component are as follows:

- {name} An NCName as defined by [*XML Namespaces [p.46]*].
- {target namespace} A namespace name, as defined in [*XML Namespaces [p.46]*].
- {parts} A set of named part definitions.

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

2.2.2 XML Representation of Message Component

The XML representation for a message definition component is an *element information item* with the following Infoset properties;

- A [local name] of `message`
- A [namespace name] of "http://www.w3.org/2003/03/wsdl"
- One or more *attribute information item*s amongst its [attributes] as follows;
  - A `name` *attribute information item* as described below in **2.2.2.1 name attribute information item with message [owner]** [p.11] .
  - 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/2003/03/wsdl".
- Zero or more *element information item*s amongst its [children], in order as follows;
  - An optional `documentation` *element information item* (see **5. Documentation** [p.38]).
  - Zero or more `part` *element information item*s (see **2.3.2 XML Representation of Part Component** [p.12])
- Zero or more namespace qualified *element information item*s amongst its [children]. Such *element information item*s MUST be a member of one of the element substitution groups related to messages described in **6. Language Extensibility** [p.38] .

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

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

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*.

2.2.3 Mapping Message's XML Representation to Component Properties

The mapping between the properties of the Message Component (see **2.2.1 The Message Component** [p.9]) and the XML Representation of the `message` *element information item* (see **2.2.2 XML Representation of Message Component** [p.10]) is as described in **Table 3** [p.11] .

Table 3: Mapping between Message 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*
{parts}	The set of part definitions corresponding to the `part` *element information item*s in [children], if any.

2.3 Part

2.3.1 The Part Component

A part component describes a portion of a particular message that a web service sends or receives. The format of a part is described by reference to type definition or element declaration components. Such components may be drawn from any appropriate type system. However WSDL makes special recognition of the XML Schema language [*XML Schema: Structures [p.46]*],[*XML Schema: Datatypes [p.46]*] as the RECOMMENDED type specification language (see **3. Types** [p.33]).

Part components are local to a given message component, they cannot be referred to by QName.

The properties of the Part Component are as follows:

- {name} An NCName as defined by [*XML Namespaces [p.46]*].

- {content reference} A reference to either an element declaration or a type definition component

For each part component in the {parts} property of a given message component the {name} property MUST be unique.

2.3.2 XML Representation of Part Component

The XML representation for a part definition is an *element information item* with the following Infoset properties;

- A [local name] of `part`
- A [namespace name] of "http://www.w3.org/2003/03/wsdl"
- Two or more *attribute information item*s amongst its [attributes] as follows;
  - A `name` *attribute information item* as described below in **2.3.2.1 name attribute information item with part [owner]** [p.13] .
  - One of the following;
    - An `element` *attribute information item* as described below in **2.3.2.2 element attribute information item** [p.13] .
    - A `type` *attribute information item* as described below in **2.3.2.3 type attribute information item** [p.13] .
    - A namespace qualified *attribute information item* in a namespace other than "http://www.w3.org/2003/03/wsdl" indicating the validation rules for this part in an alternative schema language.
  - 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/2003/03/wsdl".
- Zero or more *element information item*s amongst its [children] in order, as follows;
  - An optional `documentation` *element information item* (see **5. Documentation** [p.38]) amongst its [children].
  - Zero or more namespace qualified *element information item*s. Such *element information item*s MUST be a member of one of the element substitution groups related to message parts described in **6. Language Extensibility** [p.38] .

A part carries information about its type system by the reference contained in the `type` , `element` , or alternative qualified *attribute information item*. The schema language used is determined by examining the referred-to namespace and associating it with an imported or embedded schema. WSDL uses the `type` and `element` *attribute information item*s to refer to constructs in XML Schema [*XML Schema: Structures [p.46]*],[*XML Schema: Datatypes [p.46]*]. Alternative schema languages may re-use these *attribute information item*s if appropriate. If the concepts embodied by the `type` and `element` *attribute*

information item are not appropriate for an alternative schema language, that language should specify an alternative *attribute information item* with prescribed semantics. Such *attribute information item*s are not defined by this specification.

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

The `name` *attribute information item* identifies a given `part` *element information item* inside a given `message` *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 `element` *attribute information item*

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

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

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

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

2.3.2.3 `type` *attribute information item*

The `type` *attribute information item* refers, by QName, to a type description component.

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

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

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

2.3.3 Mapping Part's XML Representation to Component Properties

The mapping between the properties of the Part Component (see **2.3.1 The Part Component** [p.11]) and the XML Representation of the `part` *element information item* (see **2.3.2 XML Representation of Part Component** [p.12]) is as described in **Table 4** [p.13] .

Table 4: Mapping between Part Component Properties and XML Representation

Property	Mapping
{name}	The actual value of the `name` *attribute information item*
{content reference}	The element declaration resolved to by the value of the `element` *attribute information item* if present, otherwise the type definition resolved to by the value of the `type` *attribute information item* if present, otherwise a similar construct in some type system as referred to by some other *attribute information item*

2.4 Port Type

2.4.1 The Port Type Component

A port type component describes a set 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, a port type is a set of operations.

A port type can optionally extend one or more other port types. In such cases the port type contains the operations of the port types it extends, along with any operations it defines.

Port types are named constructs and can be referred to by QName (see **2.14 QName resolution** [p.32]). For instance, binding components refer to port types in this way.

The properties of the Port Type Component are as follows:

- {name} An NCName as defined by [*XML Namespaces [p.46]*].
- {target namespace} A namespace name, as defined in [*XML Namespaces [p.46]*].
- {extended port types} A set of named port type definitions.
- {operations} A set of named port type operation definitions.

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

2.4.2 XML Representation of Port Type Component

The XML representation for a port type definition component is an *element information item* with the following Infoset properties;

- A [local name] of `portType`
- A [namespace name] of "http://www.w3.org/2003/03/wsdl"

- One or more *attribute information item*s amongst its [attributes] as follows;
  - A `name` *attribute information item* as described below in **2.4.2.1 name attribute information item with portType [owner]** [p.15] .
  - An optional `extends` *attribute information item* as described below in **2.4.2.2 extends attribute information item** [p.15] .
  - 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/2003/03/wsdl".
- Zero or more *element information item*s amongst its [children], as follows;
  - An optional `documentation` *element information item* (see **5. Documentation** [p.38]).
  - Zero or more `operation` *element information item*s **2.5.2 XML Representation of Port Type Operation Component** [p.17] .
- Zero or more namespace qualified *element information item*s amongst its [children]. Such *element information item*s MUST be a member of one of the element substitution groups related to port types described in **6. Language Extensibility** [p.38] .

2.4.2.1 `name` *attribute information item* with `portType` [owner]

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

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*.

2.4.2.2 `extends` *attribute information item*

The `extends` *attribute information item* lists the port types that this port type 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*.

2.4.3 Mapping Port Type's XML Representation to Component Properties

The mapping between the properties of the Port Type Component (see **2.4.1 The Port Type Component** [p.14]) and the XML Representation of the `portType` *element information item* (see **2.4.2 XML Representation of Port Type Component** [p.14]) is as described in **Table 5** [p.16] .

Table 5: Mapping between Port Type 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*
{extended port types}	The set of port type definitions resolved to by the values in the `extends` *attribute information item* if any, plus the set of port type definitions in the {extended port types} property of those port type definitions, otherwise empty.
{operations}	The set of port type operation definitions corresponding to the `operation` *element information item*s in [children], if any, plus the set of port type operation definitions in the {operations} property of the port type definitions in {extended port types}, if any.

2.5 Port Type Operation

2.5.1 The Port Type Operation Component

A port type operation component describes an operation that a given port type supports. An operation is a set of message references. Message references may be to messages this operation accepts, that is input messages, or messages this operation sends, that is output or fault messages.

Port type operation components are local to port type components, they cannot be referred to by QName, despite having both {name} and {target namespace} properties

The properties of the Port Type Operation Component are as follows:

- {name} An NCName as defined by [*XML Namespaces [p.46]*].
- {target namespace} A namespace name, as defined in [*XML Namespaces [p.46]*].
- {variety} One of Input-Only, Output-Only, Input-Output or Output-Input
- {messages} A set of message reference components

For each port type operation component in the {operations} property of a port type component the combination of {name} and {target namespace} properties must be unique.

In cases where, due to a port type extending one or more other port types, two or more port type operation components have the same value for their {name} and {target namespace} properties, then the component models of those port type operation components MUST be equivalent (see **2.12 Equivalence of components** [p.32]). If the port type operation components are equivalent then they are considered to collapse into a single component. It is an error if two port type 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 port types 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 port types cannot both form part of the derivation chain of a derived port type unless those operations are the same operation. Therefore it is considered good practice to ensure, where necessary, that operation names within a namespace are unique, thus allowing such derivation to occur without error.

2.5.1.1 Port Type Operation Varieties

The following list defines the semantics of each of the possible values of the {variety} property:

- {variety} is Input-Output. The semantics are that when the input message is sent to the service, one of the following MUST happen: the output message is generated; or one of the fault messages listed is generated instead. It is an error for both an output message and a fault message to be generated in response to the same input message.
- {variety} is Input-Only. The semantics are that when a message is sent to the service, the service consumes it but does not produce any output message. There MUST NOT be any fault messages indicated in this case.
- {variety} is Output-Input. The semantics are that the service will generate the output message and in return the input message MUST be received or one of the fault messages listed MUST be received.
- {variety} is Output-Only. The semantics are that the service will generate the output message but does not expect to receive any response message or fault messages. There MUST NOT be any fault messages indicated in this case.

2.5.2 XML Representation of Port Type Operation Component

The XML representation for a port type 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/2003/03/wsdl"
- One or more *attribute information item*s amongst its [attributes] as follows;

    - A `name` *attribute information item* as described below in **2.5.2.1 name attribute information item with operation [owner]** [p.18] .
    - 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/2003/03/wsdl".
- One or more *element information item* amongst its [children] as follows:
    - An optional `documentation` *element information item* (see **5. Documentation** [p.38]).
    - One of:
        - An `input` *element information item* followed by an optional `output` *element information item* and zero or more `fault` *element information item*s
        - An `output` *element information item* followed by an optional `input` *element information item* and zero or more `fault` *element information item*s
- Zero or more namespace qualified *element information item*s amongst its [children]. Such *element information item*s MUST be a member of one of the element substitution groups related to port type operations described in **6. Language Extensibility** [p.38] .

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

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

2.5.3 Mapping Port Type Operation's XML Representation to Component Properties

The mapping between the properties of the Port Type Operation Component (see **2.5.1 The Port Type Operation Component** [p.16]) and the XML Representation of the `portType` *element information item* (see **2.5.2 XML Representation of Port Type Operation Component** [p.17]) is as described in **Table 6** [p.18] .

Table 6: Mapping between Port Type 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] `portType` *element information item*.
{variety}	If [children] contains an `input` *element information item* and no `output` *element information item* then Input-Only, else if [children] contains an `input` *element information item* followed by an `output` *element information item* then Input-Output, else if [children] contains an `output` *element information item* and no `input` *element information item* then Output-Only else if [children] contains an `output` *element information item* followed by an `input` *element information item* then Output-Input.
{messages}	The set of message references corresponding to the `input` , `output` and `fault` *element information item*s in [children], if any.

2.6 Message Reference

2.6.1 The Message Reference Component

A message reference component refers to a named message that forms part of an operation.

Message reference components are local to a given port type operation component, they cannot be referred to by QName

The properties of the Message Reference Component are as follows:

- {name} An NCName as defined by [*XML Namespaces [p.46]*].
- {variety} One of input, output or fault.
- {message} A reference, by QName (see **2.14 QName resolution** [p.32]), to a named message component.

For each message reference component in the {messages} property of a port type operation component the {name} property must be unique.

2.6.2 XML Representation of Message Reference Component

The XML representation for a message reference component is an *element information item* with the following Infoset properties;

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

- A [namespace name] of "http://www.w3.org/2003/03/wsdl"
- One or more *attribute information item*s amongst its [attributes] as follows;
  - An optional `name` *attribute information item* as described below in **2.6.2.1 name attribute information item with input, output or fault [owner]** [p.20] .
  - A `message` *attribute information item* as described below in **2.6.2.2 message attribute information item with input, output or fault [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/2003/03/wsdl".
- Zero or more *element information item*s amongst its [children], in order, as follows;
  - An optional `documentation` *element information item* (see **5. Documentation** [p.38]).
  - Zero or more namespace qualified *element information item*s. Such *element information item*s MUST be a member of one of the element substitution groups related to input, output or fault children of port type operations described in **6. Language Extensibility** [p.38] .

2.6.2.1 `name` *attribute information item* with `input` , `output` or `fault` [owner]

The `name` *attribute information item* identifies a given message reference *element information item* within a given `operation` *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*.

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

The `message` *attribute information item* refers, by QName, to a message 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*.

2.6.3 Mapping Message Reference's XML Representation to Component Properties

The mapping between the properties of the Message Reference Component (see **2.6.1 The Message Reference Component** [p.19]) and the XML Representation of the message reference *element information item* (see **2.6.2 XML Representation of Message Reference Component** [p.19]) is as described in **Table 7** [p.21] .

Table 7: Mapping between Message Reference Component Properties and XML Representation

Property	Mapping
{name}	The actual value of the `name` *attribute information item* if present, otherwise if {variety} is input or output and the {variety} of the enclosing Port Type Operation component is Input-Only or Output-Only then the actual value of the `name` *attribute information item* on the enclosing `operation` *element information item*, otherwise if {variety} is input and the {variety} of the enclosing Port Type Operation component is Input-Output then the actual value of the `name` *attribute information item* on the enclosing `operation` with 'Request' appended, otherwise if {variety} is output and the {variety} of the enclosing Port Type Operation component is Input-Output or {variety} is input and the {variety} of the enclosing Port Type Operation component is Output-Input then the actual value of the `name` *attribute information item* on the enclosing `operation` with 'Response' appended, otherwise if {variety} is output and the {variety} of the enclosing Port Type Operation component is Output-Input then the actual value of the `name` *attribute information item* on the enclosing `operation` with 'Solicit' appended.
{variety}	If the [local name] of the *element information item* is `input` then input, else if the [local name] of the *element information item* is `output` then output, else if the [local name] of the *element information item* is `fault` then fault.
{message}	The message component resolved to by the value of the `message` *attribute information item*

2.7 Binding

Editorial note: Should WSDL 1.2 (part 2) including bindings for SOAP 1.1?	June 25, 2002
The WS Description WG currently investigating whether it will support SOAP 1.1 in this specification, as a W3C note, or otherwise. We will of course support SOAP 1.2.	

2.7.1 The Binding Component

A binding component described a concrete binding of a port type component and associated operations to a particular concrete message format and transmission protocol.

No concrete binding details are given in this specification. The companion specification, *WSDL (Version 1.2): Bindings* [*WSDL 1.2 Bindings [p.46]*] defines such bindings for SOAP 1.1 [*SOAP 1.1 [p.47]*], SOAP 1.2 [*SOAP 1.2 Part 1: Messaging Framework [p.47]*], HTTP [*IETF RFC 2616 [p.47]*] and MIME [*IETF RFC 2045 [p.47]*]. 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 **2.14 QName resolution** [p.32]). For instance, service 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.46]*].
- {target namespace} A namespace name, as defined in [*XML Namespaces [p.46]*].
- {port type} A named port type definition
- {operations} A set of named binding operation definitions

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

2.7.2 XML Representation of Binding Component

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/2003/03/wsdl"
- One or more *attribute information item*s amongst its [attributes] as follows;
    - A `name` *attribute information item* as described below in **2.7.2.1 name attribute information item with binding [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/2003/03/wsdl".
- Zero or more *element information item*s amongst its [children], as follows;
    - An optional `documentation` *element information item* (see **5. Documentation** [p.38]).
    - Zero or more `operation` *element information item*s (see **2.8.2 XML Representation of Binding Operation Component** [p.24]).

- Zero or more namespace qualified *element information item*s amongst its [children]. Such *element information item*s MUST be a member of one of the element substitution groups related to bindings described in **6. Language Extensibility** [p.38] . Such *element information item*s are considered to be binding extension elements (see **2.7.2.3 Binding extension elements** [p.23]).

2.7.2.1 `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*.

2.7.2.2 `type` *attribute information item* with `binding` [owner]

The `type` *attribute information item* refers, by QName, to a port type component

The `type` *attribute information item* has the following Infoset properties;

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

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

2.7.2.3 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.

2.7.3 Mapping Binding's XML Representation to Component Properties

The mapping between the properties of the Binding Component (see **2.7.1 The Binding Component** [p.21]) and the XML Representation of the `binding` *element information item* (see **2.7.2 XML Representation of Binding Component** [p.22]) is as described in **Table 8** [p.23] .

Table 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*
{port type}	The port type definition resolved to by the actual value of the `type` *attribute information item.*
{operations}	The set of binding operation definitions corresponding to the `operation` *element information item*s in [children], if any.

2.8 Binding Operation

2.8.1 The Binding Operation Component

A binding operation component describes a concrete binding for a particular operation of a port type to a particular concrete message format.

Binding operation components are local to a given binding component

The properties of the Binding Operation Component are as follows:

- {name} An NCName as defined by [*XML Namespaces [p.46]*].
- {target namespace} A namespace name, as defined in [*XML Namespaces [p.46]*].
- {messages} A set of binding message reference components

For each binding operation component in the {operations} property of a binding component the combination of {name} and {target namespace} properties must be unique.

2.8.2 XML Representation of Binding Operation Component

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/2003/03/wsdl"
- One or more *attribute information item*s amongst its [attributes] as follows;
    - A `name` *attribute information item* as described below in **2.8.2.1 name attribute information item with operation [owner]** [p.25] .

    - 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/2003/03/wsdl".

- Zero or more *element information item*s amongst its [children], in order, as follows;
    - An optional `documentation` *element information item* (see **5. Documentation** [p.38]).
    - One of;
        - An `input` *element information item* followed by an optional `output` *element information item*
        - An `output` *element information item* followed by an optional `input` *element information item*
    - Zero or more `fault` *element information item*s
- Zero or more namespace qualified *element information item*s amongst its [children]. Such *element information item*s MUST be a member of one of the element substitution groups related to binding operations described in **6. Language Extensibility** [p.38] . Such *element information item*s are considered to be binding operation extension elements as described below (see **2.8.2.2 Binding operation extension elements** [p.25]).

2.8.2.1 `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:NCName*.

2.8.2.2 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.

2.8.3 Mapping Binding Operation's XML Representation to Component Properties

The mapping between the properties of the Binding Operation Component (see **2.8.1 The Binding Operation Component** [p.24]) and the XML Representation of the `binding` *element information item* (see **2.8.2 XML Representation of Binding Operation Component** [p.24]) is as described in **Table 9** [p.26] .

Table 9: Mapping between Binding 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] `portType` *element information item*.
{messages}	The set of binding message reference components corresponding to the `input` , `output` and `fault` *element information item*s in [children], if any.

2.9 Binding Message Reference

2.9.1 The Binding Message Reference Component

A binding message reference component describes a concrete binding for a particular message in an operation to a particular concrete message format.

Binding message reference components are local to a given binding operation component, they cannot be referred to by QName.

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

- {name} An NCName as defined by [*XML Namespaces [p.46]*].
- {variety} One of input, output or fault

For each binding message reference component in the {messages} property of a binding operation component the {name} property must be unique.

2.9.2 XML Representation of Binding Message Reference Component

The XML representation for a binding message reference component is an *element information item* with the following Infoset properties;

- A [local name] of `input` , `output` or `fault` .
- A [namespace name] of "http://www.w3.org/2003/03/wsdl"
- Zero or more *attribute information item*s amongst its [attributes] as follows;
    - An optional `name` *attribute information item* as described below in **2.9.2.1 name attribute information item with input, output or fault [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/2003/03/wsdl".

- Zero or more *element information item*s amongst its [children], in order, as follows;
  - An optional `documentation` *element information item* (see **5. Documentation** [p.38]).
  - Zero or more namespace qualified *element information item*s. Such *element information item*s MUST be a member of one of the element substitution groups related to input, output or fault children of binding operations described in **6. Language Extensibility** [p.38] . Such *element information item*s are considered to be binding message reference extension elements, as described below (see **2.9.2.2 Binding message reference extension elements** [p.27]).

2.9.2.1 `name` *attribute information item* with `input` , `output` or `fault` [owner]

The `name` *attribute information item* identifies a given binding message reference *element information item* within a given `operation` *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*.

2.9.2.2 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.

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

The mapping between the properties of the Binding Message Reference Component (see **2.9.1 The Binding Message Reference Component** [p.26]) and the XML Representation of the `binding` *element information item* (see **2.9.2 XML Representation of Binding Message Reference Component** [p.26]) is as described in **Table 10** [p.27] .

Table 10: Mapping between Binding Message Reference Component Properties and XML Representation

Property	Mapping
{name}	The actual value of the `name` *attribute information item* if present, otherwise the actual value of the `name` *attribute information item* on the *element information item* with the same [local name] whose [parent] `operation` *element information item* has the same value for its `name` *attribute information item* as the `operation` *element information item* in the `binding` *element information item* but is rather in the `portType` *element information item* referred to by the `type` *attribute information item*.
{variety}	If the [local name] of the *element information item* is `input` then input, else if the [local name] of the *element information item* is `output` then output, else if the [local name] of the *element information item* is `fault` then fault.

2.10 Service

2.10.1 The Service Component

A service component describes the set of port types that a service provides and the ports they are provided over.

Service are named constructs and can be referred to by QName (see **2.14 QName resolution** [p.32]).

The properties of the Service Component are as follows:

- {name} An NCName as defined by [*XML Namespaces [p.46]*].
- {target namespace} A namespace name, as defined in [*XML Namespaces [p.46]*].
- {port types} A set of port type components
- {ports} A set of port components

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

2.10.2 XML Representation of Service Component

The XML representation for a service definition component is an *element information item* with the following Infoset properties;

- A [local name] of `service`
- A [namespace name] of "http://www.w3.org/2003/03/wsdl"

- One or more *attribute information item*s amongst its [attributes] as follows;
    - A `name` *attribute information item* as described below in **2.10.2.1 name attribute information item with service [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/2003/03/wsdl".
- One or more *element information item* amongst its [children], as follows;
    - An optional `documentation` *element information item* (see **5. Documentation** [p.38]).
    - One or more `port` *element information item*s (see **2.11.2 XML Representation of Port Component** [p.30]
- Zero or more namespace qualified *element information item*s amongst its [children]. Such *element information item*s MUST be a member of one of the element substitution groups related to services described in **6. Language Extensibility** [p.38] .

2.10.2.1 `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*.

2.10.3 Mapping Service's XML Representation to Component Properties

The mapping between the properties of the Service Component (see **2.10.1 The Service Component** [p.28]) and the XML Representation of the `service` *element information item* (see **2.10.2 XML Representation of Service Component** [p.28]) is as described in **Table 11** [p.29] .

Table 11: 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*
{port type}	The set of port type components resolved to by the actual value of the `portType` *attribute information item*s on the bindings resolved to by the `binding` *attribute information item*s on the `port` *element information item*s in [children], if any.
{ports}	The port components corresponding to the `port` *element information item*s in [children] if any.

2.11 Port

2.11.1 The Port Component

A port component defines the particulars of a specific end-point at which a given service is available.

Port components are local to a given service component, they cannot be referred to by QName

The properties of the Port Component are as follows:

- {name} An NCName as defined by [*XML Namespaces [p.46]*].
- {binding} A named binding component

For each port component in the {ports} property of a service component the {name} property must be unique.

2.11.2 XML Representation of Port Component

The XML representation for a port definition component is an *element information item* with the following Infoset properties;

- A [local name] of `port`
- A [namespace name] of "http://www.w3.org/2003/03/wsdl"
- Two or more *attribute information item*s amongst its [attributes] as follows;
    - A `name` *attribute information item* as described below in **2.11.2.1 name attribute information item with port [owner]** [p.31] .

    - A `binding` *attribute information item* as described below in **2.11.2.2 binding attribute information item with port [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/2003/03/wsdl".

- Zero or more *element information item* amongst its [children], as follows;

    - An optional `documentation` *element information item* (see **5. Documentation** [p.38]).

    - Zero or more namespace qualified *element information item*s. Such *element information item*s MUST be a member of one of the element substitution groups related to ports described in **6. Language Extensibility** [p.38] . Such *element information item*s are considered to be port extension elements (see **2.11.2.3 Port extension elements** [p.31]).

2.11.2.1 `name` *attribute information item* with `port` [owner]

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

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 `binding` *attribute information item* with `port` [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*.

2.11.2.3 Port extension elements

Port extension elements are used to provide information specific to a particular port 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 port component with additional properties and specify the mapping between those properties and the XML representation.

2.11.3 Mapping Port's XML Representation to Component Properties

The mapping between the properties of the Port Component (see **2.11.1 The Port Component** [p.30]) and the XML Representation of the `port` *element information item* (see **2.11.2 XML Representation of Port Component** [p.30]) is as described in **Table 12** [p.32] .

Table 12: Mapping between Port 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*

2.12 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 (messages, port types, 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.

2.13 Symbol Spaces

This specification defines 4 symbol spaces, one for each top-level component type (message, port type, 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 a port type 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.46]*]. Other type systems may define additional symbol spaces.

2.14 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 a port type 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 a port type (as referred to by the `portType` *attribute information item* on a binding), the {port types} 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

At the abstract level, the Types Component is a container for 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.46]*],[*XML Schema: Datatypes [p.46]*] 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 schemas defined or referenced in the Types Component provide the type system used for message parts. Message parts indicate their structure and content by using the standard *attribute information item*s `type` and `element` , or for alternate schema languages in which these concepts do not map well, by defining alternative *attribute information item* extensions to the `part` *element information item*.

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/2003/03/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/2003/03/wsdl
- Zero or more *element information item*s amongst its [children] as follows;
    - An optional `documentation` *element information item* (see **5. Documentation** [p.38]) 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/2003/03/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.

3.1.1 Importing XML Schema

Importing an XML Schema uses the `xs:import` mechanism defined by XML Schema[*XML Schema: Structures [p.46]*],[*XML Schema: Datatypes [p.46]*], with its syntax and semantics, plus some restrictions. 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 `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 and element definitions imported from the referenced schema. If the referenced schema contains a `targetNamespace` *attribute information item* on its `xs:schema` *element information item*, then these values of these two *attribute information item*s MUST be identical. If the schema does not have a `targetNamespace` *attribute information item*, then the namespace specified by the `namespace` *attribute information item* is applied to all components of the schema as if it contained a corresponding targetNamespace declaration. 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] that 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 Component. 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 `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 and element definitions 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] that has no value.

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

3.1.3 References to Type and Element Definitions

Whether embedded or imported, the type and element definitions present in a schema may be referenced from a message part.

A named, global `xs:simpleType` or `xs:complexType` definition may be referenced from the `part` *element information item* using the `type` *attribute information item*. The QName is constructed from the `targetNamespace` of the schema and the content of the `name` *attribute information item* of the `simpleType` or `complexType` *element information item*. A `type` *attribute information item* may not be used to refer to an `xs:element` *element information item*.

A named, global `xs:element` definition may be referenced from the `part` *element information item* using the `element` *attribute 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*. An `element` *attribute information item* may not be used to refer to an `xs:simpleType` or `xs: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 parts 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 parts. 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 for inclusion in the Types Component, which references, names, and locates the schema instance (an "import" *element information item*). See **E. Examples of Specifications of Extension Elements for Alternative Schema Language Support.** [p.54] for examples of using other schema languages.

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

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.46]*], 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/2003/03/wsdl".
- An *attribute information item* which has:
  - 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

The WSDL `import` *element information item*, like the `include` *element information item* (see **4.1 Including Descriptions** [p.36]) 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.46]*], 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:

- A [local name] of `import` .
- A [namespace name] of "http://www.w3.org/2003/03/wsdl".
- An *attribute information item* which has:
  - 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 can 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.*

- An OPTIONAL *attribute information item* which has:
  - 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

WSDL uses the optional `documentation` *element information item* as a container for human readable 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.46]*]). 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/2003/03/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 two extensibility models, one based on element substitution groups and another based on qualified attributes.

6.1 Element based extensibility

WSDL allows extensions to be defined in terms of *element information item*s. Several abstract global element declarations serve as the heads of substitution groups. Some are specific to a particular point in the schema while others appear in multiple places. This allows extension to be specific to a particular point of the WSDL schema or more general. **Table 13** [p.39] shows the complete list of element substitution groups that along with their allowed locations.

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 Base type for extensibility elements** [p.45] 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/2003/03/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 portTypes and operations only, and will consequently skip bindings.

Table 13: Element substitution groups defined for element based extensibility

Element name	Allowed in [children] property of	Notes

globalExt	• `wsdl:definitions` after `wsdl:import` , `wsdl:include` and `wsdl:types` *element information item*s if present. • `wsdl:message` • `wsdl:part` • `wsdl:portType` • `wsdl:operation` • `wsdl:input` • `wsdl:output` • `wsdl:fault` • `wsdl:binding` • `wsdl:service` • `wsdl:port`	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as a child of any *element information item* in the "http://www.w3.org/2003/03/wsdl" namespace with the exception of `wsdl:import` , `wsdl:include` and `wsdl:types` .
msgAndportTypeExt	• `wsdl:message` • `wsdl:portType`	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as a child of `wsdl:message` and/or `wsdl:portType`
msgportTypeAndBindingExt	• `wsdl:message` • `wsdl:portType` • `wsdl:binding`	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as a child of `wsdl:message` , `wsdl:portType` and/or `wsdl:binding`
msgportTypeBindingServiceExt	• `wsdl:message` • `wsdl:portType` • `wsdl:binding` • `wsdl:service`	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as a child of `wsdl:message` , `wsdl:portType` , `wsdl:binding` and/or `wsdl:service`

bindingAndServiceExt	• `wsdl:binding` • `wsdl:service`	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as a child of `wsdl:binding` and/or `wsdl:service`
preImportInclude	• `wsdl:definitions` before `wsdl:import` and `wsdl:include` , if present.	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear before the `wsdl:import` and `wsdl:include` *element information item*s
inImportInclude	• `wsdl:definitions` before, between or after `wsdl:import` and `wsdl:include` , if present.	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear interleaved with the `wsdl:import` and `wsdl:include` *element information item*s
preTypes	• `wsdl:definitions` after `wsdl:import` and `wsdl:include` and before `wsdl:types` , if present.	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear after the `wsdl:import` and `wsdl:include` *element information item*s but before the `wsdl:types` *element information item*
postTypes	• `wsdl:definitions` after `wsdl:types` , if present.	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear after the `wsdl:types` *element information item* (potentially interleaved with `wsdl:message` , `wsdl:portType` , `wsdl:/binding` and `wsdl:service` *element information item*s.
Abstract elements related to message		
msgExt	• `wsdl:message`	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:message`

msgPrePartExt	• `wsdl:message` before `wsdl:part` *element information item*s, if present.	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:message` before `wsdl:part`
msgPostPartExt	• `wsdl:message` after `wsdl:part` *element information item*s, if present.	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:message` after `wsdl:part`
msgPartExt	• `wsdl:part`	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:message/wsdl:part`
Abstract elements related to portType		
portTypeExt	• `wsdl:portType`	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:portType`
portTypePreOpExt	• `wsdl:portType` before `wsdl:operation` *element information item*s, if present.	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:portType` before `wsdl:operation`
portTypePostOpExt	• `wsdl:portType` after `wsdl:operation` *element information item*s, if present.	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:portType` after `wsdl:operation`
portTypeOpExt	• `wsdl:operation` with `wsdl:portType` [parent]	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:operation` when it is a child of `wsdl:portType`

portTypeOpMsgExt	• `wsdl:input` with `wsdl:operation` [parent] which itself has `wsdl:portType` [parent] • `wsdl:output` with `wsdl:operation` [parent] which itself has `wsdl:portType` [parent]	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:input` or `wsdl:output` when they appear as children of `wsdl:operation` which itself appears as a child of `wsdl:portType`
portTypeOpFaultExt	• `wsdl:fault` with `wsdl:operation` [parent] which itself has `wsdl:portType` [parent]	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:fault` when it appears as a child of `wsdl:operation` which itself appears as a child of `wsdl:portType`
Abstract elements related to bindings		
bindingExt	• `wsdl:binding`	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:binding`
bindingPreOpExt	• `wsdl:binding` before `wsdl:operation` *element information item*s, if present.	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:binding` before `wsdl:operation`
bindingPostOpExt	• `wsdl:binding` after `wsdl:operation` *element information item*s, if present.	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:binding` after `wsdl:operation`

bindingOpExt	• `wsdl:operation` with `wsdl:binding` [parent]	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:operation` appearing as children of `wsdl:binding`
bindingOpMsgExt	• `wsdl:input` with `wsdl:operation` [parent] which itself has `wsdl:binding` [parent] • `wsdl:output` with `wsdl:operation` [parent] which itself has `wsdl:binding` [parent]	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:input` or `wsdl:output` when they appear as children of `wsdl:operation` which itself appears as a child of `wsdl:binding`
bindingOpFaultExt	• `wsdl:fault` with `wsdl:operation` [parent] which itself has `wsdl:binding` [parent]	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:fault` when it appears as a child of `wsdl:operation` which itself appears as a child of `wsdl:binding`
Abstract elements related to service		
serviceExt	• `wsdl:service`	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:service`
servicePrePortExt	• `wsdl:service` before `wsdl:port` *element information item*s, if present.	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:service` before `wsdl:port`
servicePostPortExt	• `wsdl:service` after `wsdl:port` *element information item*s, if present.	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:service` after `wsdl:port`

portExt	• `wsdl:port`	This *element information item* serves as the head of a substitution group for extensibility elements that wish to appear as children of `wsdl:port`

Example 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.45]) 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/2003/03/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/2003/03/wsdl" to appear on any *element information item* whose namespace name IS "http://www.w3.org/2003/03/wsdl". Such *attribute information item*ss can be used to annotate WSDL constructs such as messages, port types 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 1.2 Bindings]
Web Services Description (WSDL) Version 1.2: Bindings, J-J. Moreau, J. Schlimmer, Editors. World Wide Web Consortium, 24 January 2003. This version of the Web Services Description Version 1.2:

Bindings Specification is available is available at http://www.w3.org/TR/2003/WD-wsdl12-bindings-20030124. The latest version of Web Services Description Version 1.2: Bindings is available at http://www.w3.org/TR/wsdl12-bindings.

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, 19 December 2002. This version of the SOAP Version 1.2 Part 1 Specification is http://www.w3.org/TR/2002/CR-soap12-part1-20021219/. The latest version of SOAP Version 1.2 Part 1 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 Note is http://www.w3.org/TR/2001/NOTE-wsdl-20010315. The latest version of Web Services Description Language is available at http://www.w3.org/TR/wsdl.

[WSDL 1.2 Primer]
Web Services Description (WSDL) Version 1.2: Primer, K. Sankar, K. Liu, D. Booth, Editors. World Wide Web Consortium, 3 March 2003. The editors' version of the Web Services Description Version 1.2: 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, 29 April 2002. This version of the Web Services Description Requirements document is http://www.w3.org/TR/2002/WD-ws-desc-reqs-20020429. 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/2002/PR-xptr-framework-20021113 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 1.2 documents serialized as XML. It is referenced by the corresponding IANA registration document [*WSDL MediaType [p.46]*].

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.46]*].

Encoding considerations:

Identical to those of "application/xml" as described in [*RFC 3023 [p.46]*], section 3.2, as applied to the WSDL document infoset.

Security considerations:

See section **A.2 Security considerations** [p.50] .

Interoperability considerations:

There are no known interoperability issues.

Published specification:

This document (Part 1) and [*WSDL 1.2 Bindings [p.46]*] (Part 2).

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.46]*], section 5 or the syntax defined in **C.2 Fragment Identifiers** [p.51] .

Base URI:

As specified in [*RFC 3023 [p.46]*], section 6.

Macintosh File Type code:

TEXT

Person and email address to contact for further information:

@@@ <@@@@>

Intended usage:

COMMON

Author/Change controller:

The WSDL 1.2 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.46]*], 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), Allen Brookes (Rogue Wave Softwave), Amelia Lewis (TIBCO/Extensibility), Arthur Ryman (IBM), Barbara Zengler (DaimlerChrysler Research and Technology), Dale Moberg (Cyclone Commerce), Dan Kulp (IONA Technologies), Daniel Schutzer (Citigroup), Dave Solo (Citigroup), David Booth (W3C), Dietmar Gaertner (Software AG), Don Mullen (TIBCO Software), Erik Ackerman (Lexmark), Glen Daniels (Macromedia), Igor Sedukhin (Computer Associates), Jacek Kopecky (Systinet), Jean-Jacques Moreau (Canon), Jeff Mischkinsky (Oracle Corporation), Jeffrey Schlimmer (Microsoft Corporation), Jerry Thrasher (Lexmark), Johan Pauhlsson (L'Échangeur), Jonathan Marsh (*Chair*, Microsoft Corporation), Kevin Canyang Liu (SAP), Laurent De Teneuille (L'Échangeur), 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), Philippe Le Hégaret (W3C), Prasad Yendluri (webMethods, Inc.), Roberto Chinnici (Sun Microsystems), Sandeep Kumar (Cisco Systems), Sandra Swearingen (U.S. Department of Defense, U.S. Air Force), Sanjiva Weerawarana (IBM), Stefano Pogliani (Sun Microsystems), Stephen White (SeeBeyond), Steve Graham (Global Grid Forum), Steve Lind (AT&T), Steve Tuecke (Global Grid Forum), Tim Finin (University of Maryland), Tom Jordahl (Macromedia), Umit Yalcinalp (Oracle Corporation), Waqar Sadiq (Electronic Data Systems), William Stumbo (Xerox), William Vambenepe (Hewlett-Packard Company), Youenn Fablet (Canon)

Previous members were: Aaron Skonnard (DevelopMentor), Don Wright (Lexmark), Jochen Ruetschlin (DaimlerChrysler Research and Technology), Joyce Yang (Oracle Corporation), Keith Ballinger (Microsoft), Krishna Sankar (Cisco Systems), Mario Jeckle (DaimlerChrysler Research and Technology), Pallavi Malu (Intel Corporation)

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

C. URI References for WSDL constructs (Non-Normative)

This appendix provides a syntax for URI references for constructs found in a WSDL document. This includes first class constructs such as message, port type, binding and service and subordinate constructs such as message parts and port type operations. The URI references are easy to understand and compare, while imposing no burden on the WSDL author.

C.1 WSDL URIs

There are two main cases for WSDL URIs:

- the URI of a WSDL document
- the URI of a WSDL namespace

The URI of a WSDL document can be dereferenced to give a resource representation that contributes component definitions to a single WSDL namespace. If the media type is set to the WSDL media type, then the fragment identifiers can be used to identify the main components that are defined in the document.

However, this appendix specifies the use of the namespace URI with the WSDL fragment identifiers to form a URI-reference.

Editorial note: MJG	20030203
The URI of a WSDL namespace may not be dereferencable, or if it is dereferencable, then it may not resolve to a resource representation that has the WSDL media type. For example, a namespace URI may resolve to an HTML document that describes the namespace. Since the namespace URI does not necessarily reference a resource with the WSDL media type, the use of WSDL fragment identifiers with it is not strictly in compliance with the definition of URI-reference. Comment from the W3C Technical Architecture Group on this point is invited.	

C.2 Fragment Identifiers

The following fragment identifier syntax is compliant with the [*XPointer Framework [p.48]*].

The URI reference for a WSDL construct is the {target namespace} property of either the construct itself, in the case of messages, port types, bindings, services or operations, or the {target namespace} property of an ancestor construct. The URI provided by the {target namespace} property is combined with a fragment identifier, where the fragment identifier is constructed from the {name} property of the construct and the {name} properties of its ancestors as a path according to **Table 14** [p.51] . In that table the first column gives the name of the WSDL construct as the [local name] of the *element information item* that represents that construct in a WSDL document. Columns two, three and four populate the variables x, y and z respectively. These variables are then used to construct the fragment in column five.

Table 14: Rules for determining fragments for WSDL constructs

Construct	x	y	z	Fragment
message	{name} property of message	n/a	n/a	`message(x)`
part	{name} property of part	{name} property of message	n/a	`part(y/x)`
portType	{name} property of port type	n/a	n/a	`portType(x)`
operation	{name} property of operation	{name} property of parent port type	n/a	`operation(y/x)`
input	{name} property of message reference	{name} property of parent operation	{name} property of grandparent port type	`input(z/y/x)`
output	{name} property of message reference	{name} property of parent operation	{name} property of grandparent port type	`output(z/y/x)`
fault	{name} property of message reference	{name} property of parent operation	{name} property of grandparent port type	`fault(z/y/x)`
binding	{name} property of binding	n/a	n/a	`binding(x)`
service	{name} property of service	n/a	n/a	`service(x)`
port	{name} property of port	{name} property of parent service	n/a	`port(y/x)`

Note that the above rules are defined in terms of component properties rather than transfer syntax. Because of the mappings defined for these properties, values are always available, even if certain optional *attribute information item*s are missing from the WSDL document. This allows meaningful URI-references to be formed for constructs like message reference components (see **2.6 Message Reference** [p.19]) whose *element information item*s <input> and <output> may not actually carry a `name` *attribute information item*.

C.3 Extension Elements

WSDL has an open content model. It is therefore possible for an extension to define new components. The XPointer Framework scheme for components added by extensions is:

```
extension(extension-namespace, extension-specific-syntax)
```

where extension-namespace is the namespace that identifies the extension, e.g. for SOAP the namespace is http://www.w3.org/2003/03/wsdl/soap12, and extension-specific-syntax is defined by the extension. The owner of the extension must define any components contributed by the extension and a syntax for identifying them.

C.4 Example

Consider the following WSDL located at http://schemas.airlines.org/TicketAgent.wsdl:

Example 2: URI References - Example WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="TicketAgent"
    targetNamespace="http://airline.wsdl/ticketagent/"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://airline.wsdl/ticketagent/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsd1="http://airline/">
    <import location="TicketAgent.xsd" namespace="http://airline/"/>
    <message name="listFlightsRequest">
        <part name="depart" type="xsd:dateTime"/>
        <part name="origin" type="xsd:string"/>
        <part name="destination" type="xsd:string"/>
    </message>
    <message name="listFlightsResponse">
        <part name="result" type="xsd1:ArrayOfString"/>
    </message>
    <message name="reserveFlightRequest">
        <part name="depart" type="xsd:dateTime"/>
        <part name="origin" type="xsd:string"/>
        <part name="destination" type="xsd:string"/>
        <part name="flight" type="xsd:string"/>
    </message>
    <message name="reserveFlightResponse">
        <part name="result" type="xsd:string"/>
    </message>
    <portType name="TicketAgent">
        <operation name="listFlights" parameterOrder="depart origin destination">
            <input message="tns:listFlightsRequest" name="listFlightsRequest"/>
            <output message="tns:listFlightsResponse" name="listFlightsResponse"/>
        </operation>
        <operation name="reserveFlight" parameterOrder="depart origin destination flight">
            <input message="tns:reserveFlightRequest" name="reserveFlightRequest"/>
            <output message="tns:reserveFlightResponse" name="reserveFlightResponse"/>
        </operation>
    </portType>
</definitions>
```

Its conceptual elements have the following URI-references:

Example 3: URI References - Example URIs

```
http://airline.wsdl/ticketagent/#message(listFlightsRequest)
http://airline.wsdl/ticketagent/#part(listFlightsRequest/depart)
http://airline.wsdl/ticketagent/#part(listFlightsRequest/origin)
http://airline.wsdl/ticketagent/#part(listFlightsRequest/destination)
http://airline.wsdl/ticketagent/#message(listFlightsResponse)
```

```
http://airline.wsdl/ticketagent/#part(listFlightsResponse/result)
http://airline.wsdl/ticketagent/#message(reserveFlightRequest)
http://airline.wsdl/ticketagent/#part(reserveFlightRequest/depart)
http://airline.wsdl/ticketagent/#part(reserveFlightRequest/origin)
http://airline.wsdl/ticketagent/#part(reserveFlightRequest/destination)
http://airline.wsdl/ticketagent/#part(reserveFlightRequest/flight)
http://airline.wsdl/ticketagent/#message(reserveFlightResponse)
http://airline.wsdl/ticketagent/#part(reserveFlightResponse/result)
http://airline.wsdl/ticketagent/#portType(TicketAgent)
http://airline.wsdl/ticketagent/#operation(TicketAgent/listFlights)
http://airline.wsdl/ticketagent/#input(TicketAgent/listFlights/listFlightsRequest)
http://airline.wsdl/ticketagent/#output(TicketAgent/listFlights/listFlightsResponse)
http://airline.wsdl/ticketagent/#operation(TicketAgent/lreserveFlight)
http://airline.wsdl/ticketagent/#input(TicketAgent/lreserveFlight/lreserveFlightRequest)
http://airline.wsdl/ticketagent/#output(TicketAgent/lreserveFlight/lreserveFlightResponse)
```

C.5 Relation to WSDL 1.1

This appendix applies to WSDL 1.2. It is does not apply to WSDL 1.1. For example, the proposed syntax does not disambiguate overloaded operation names, which are permissible in WSDL 1.1, but not in WSDL 1.2.

D. Migrating from WSDL 1.1 to WSDL 1.2 (Non-Normative)

This section will attempt to document some of the migration concerns of going from WSDL 1.1 to WSDL 1.2. We do not claim that all migration problems will be addressed here.

D.1 Operation Overloading

WSDL 1.1 supported operation overloading and WSDL 1.2 removes it. This section will provide some rationale for it and provide hints on how to work around some scenarios.

D.2 PortType Inheritance

We now have portType inheritance.

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

E.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 may be referenced only by the `element` *attribute information item*; DTDs do not have a construct corresponding to the `type` *attribute information item*.

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 item*s, as follows:
  - A `namespace` *attribute information item* as described below.
  - An optional `location` *attribute information item* as described below.

E.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] that 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).

E.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] that has no value.

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

E.1.3 References to Element Definitions

In referring to an element definition (<!ELEMENT>) from a message part, the `type` *attribute information item* may not be used. The `element` *attribute information item* must be used. Its value 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.

E.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. The following discussion supplies the prefix rng which is mapped to the URI "http://www.relaxng.org/ns/structure/1.0".

E.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 `ns` *attribute information item* as described below.
  - A `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 item*s; it may be used solely to import a schema.

E.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*.

E.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] that has no value.

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

E.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 Types Component. 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".
- An `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.

E.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] that has no value.

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

E.2.3 References to Type and Element Definitions

Whether embedded or imported, the type and element definitions present in a schema may be referenced from a message part.

A named rng:define definition may be referenced from the `part` *element information item* using the `type` *attribute information item*. 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 `define` *attribute information item*. A `type` *attribute information item* may not be used to refer to an `rng:element` *element information item.*

A named Relax NG element definition may be referenced from the `part` *element information item* using the `element` *attribute information item*. 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* may not be used to refer to an `rng:define` *element information item*.

F. Part 1 Change Log (Non-Normative)

F.1 WSDL Specification Changes

Date	Author	Description
20030217	MJG	Minor edits to wording in **2.5.1 The Port Type Operation Component** [p.16]
20030213	MJG	Added xlink nsdecl to spec element
20030213	MJG	Incorporated text from dbooths proposal on semantics, per decision 20021031
20030213	MJG	Incorporated operationname proposal.
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 **2.6.2 XML Representation of Message Reference Component** [p.19] 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 **2.5.1 The Port Type Operation Component** [p.16]
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.8]
20021127	MJG	changed the language regarding extensibility elements in **2.1.2 XML Representation of Definitions Component** [p.7] .
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