Abstract:
The components introduced by the WS Resource Framework (WSRF) address functional aspects of modeling stateful resources (such as systems resources) using Web services. WSRF uses WSDL (currently WSDL 1.1) as the form of service description. There is an outstanding need to be able to supplement the descriptive information available about a WS-Resource. In particular, the format of additional metadata information about the components of a WS-Resource must be standardized; most notably the resource properties document [WS-ResourceProperties] and information about operations in its interface,

In the realm of resource properties, the loosely coupled nature and operations for access and update of properties [WS-ResourceProperties] benefit from additional metadata. An example of this type of metadata is the modifiability and intended usage of resource properties. Perhaps even an enumeration of value restrictions on resource property elements.

This document motivates the need for such metadata and proposes an information model representing metadata that would be specifically applicable to Manageable Resources but also capable of expressing metadata about Web services and WS-Resources in general.
This document is published by this TC as a "working draft". It is possible that it may change significantly during this process, but should nonetheless provide a stable reference for discussion and early adopters' implementations.

Committee members should send comments on this specification to the wsrf@lists.oasis-open.org list. Others should subscribe to and send comments to the wsrf-comment@lists.oasis-open.org list. To subscribe, send an email message to wsrf-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the WSRF TC web page (http://www.oasis-open.org/committees/wsrf/).
Table of Contents

1 INTRODUCTION ..................................................................................................................5
  1.1 GOALS AND REQUIREMENTS .....................................................................................6
  1.1.1 Requirements .......................................................................................................6
  1.1.2 Non-Goals .............................................................................................................6
  1.2 TERMINOLOGY ...........................................................................................................6
  1.3 NAMESPACES ..............................................................................................................7

2 TERMINOLOGY AND CONCEPTS .....................................................................................8

3 EXAMPLE ............................................................................................................................9
  3.1 THE OPERATING SYSTEM portType ..........................................................................9
  3.2 OPERATING SYSTEM PROPERTIES .................................................................10
  3.2.1 Operating System Resource Property definitions ..............................................10
  3.2.2 Identification Property definitions ......................................................................10
  3.2.3 MetadataDescriptor for Identification portType ..................................................11
  3.2.4 MetadataDescriptor for OperatingSystem portType ............................................12
  3.2.5 MetadataDescriptor for WindowsXP WS-Resource .............................................13

4 LOGICAL MODEL FOR METADATA ............................................................................15

5 INFORMATION MODEL FOR WS-RESOURCE METADATA ........................................16

6 DEFINITIONS COMPONENT .............................................................................................17
  6.1 MAPPING DEFINITIONS COMPONENT PROPERTIES TO AN XML REPRESENTATION ..................................................................................................................18
  6.2 METADATA_DESCRIPTOR COMPONENTS WITHIN A DEFINITIONS COMPONENT ..................................................................................................................18

7 METADATA_DESCRIPTOR COMPONENT ........................................................................19
  7.1 MAPPING METADATA_DESCRIPTOR COMPONENT PROPERTIES TO AN XML REPRESENTATION ........................................................................................................19
  7.2 PROPERTIES COMPONENT OF A METADATA_DESCRIPTOR ...................................22
  7.3 OPERATIONS COMPONENT OF A METADATA_DESCRIPTOR ......................................22

8 PROPERTY COMPONENT ....................................................................................................23
  8.1 MAPPING PROPERTY COMPONENT PROPERTIES TO AN XML REPRESENTATION ..........................................................23
  8.2 XML SCHEMA VALUE SPACE AND [VALIDVALUES] .............................................26
  8.3 VALIDVALUES .........................................................................................................26
  8.4 VALIDVALUERANGE ...............................................................................................28
  8.4.1 Mapping ValidValuRangeComponent Properties to an XML Representation .....28
  8.5 STATICVALUES .......................................................................................................29
  8.6 PATH OF A PROPERTY COMPONENT .........................................................................29
  8.7 RECONCILING PROPERTY COMPONENTS WITH OVERLAPPING PATH ...............29
  8.7.1 Reconciling Properties with Identical Path .........................................................31
  8.7.2 Reconciling a Property with Narrower Path ......................................................31

9 OPERATION COMPONENT ................................................................................................34
  9.1 MAPPING OPERATION COMPONENT PROPERTIES TO AN XML REPRESENTATION ........................................................................................................34

wsrf-WS-ResourceMetadataDescriptor-1.0-draft-01.doc  10/21/2004
Copyright © OASIS Open 2004. All Rights Reserved.
Page 3 of 53
1 Introduction

In the WS-Resource Framework [WSRF], elements of a WS-Resource’s state are exposed to third party requestors through an XML document. The XML document associated with a WS-Resource is called a resource properties document. The resource properties document is a projection of the WS-Resource’s state (not all of the elements of a WS-Resource’s state are exposed through the resource properties document). An individual element of state contained in a resource properties document is called a resource property. Access to the resource properties document is governed by Web services operations defined in the WS-Resource Properties [WS-Resource Properties] specification. These operations generically allow for get, set and query of resource properties.

In many cases, some of the resource properties exposed through the resource properties document are not accessible through every operation defined in WS-ResourceProperties. The most common case of this is a resource property that is “read-only” implying that a requestor may not use Web services message exchanges (such as the WS-ResourceProperties SetResourceProperties operation) to change the value of the resource property. Clearly, an implementation of a WS-Resource is likely to return a fault message if a requestor attempts to change the value of a “read-only” resource property. However, in the absence of additional metadata, there is no standard means by which the requestor can determine a priori that the resource property was not modifiable.

A similar challenge is found with WSDL operation descriptions. In many cases, it is likely that certain operations will be idempotent. That is to say the action is performed exactly once, even if the operation is invoked several times. For example, an invocation of the “Stop” operation when the WS-Resource is already in the “Quiesced” state will leave the WS-Resource in the “Quiesced” state. WSDL 1.1 does not provide a standard means by which the designer of the service can inform the requestor of this constraint.

We refer to the concept of a WS-Resource Metadata Descriptor to describe a unit of metadata information associated with the interface components of a WS-Resource. We describe an information model that outlines the components of metadata and their relationships to interface description artifacts such as WSDL 1.1 portTypes and resource properties document schema definitions.

A WS-Resource Metadata Descriptor serves multiple purposes. The first is to provide additional information about the interface (operations and resource properties) of a WS-Resource. For instance, indicating whether a resource property is changeable using Web services message exchanges such as the SetResourceProperties operation described in the WS-ResourceProperties specification [WS-ResourceProperties]. This aspect of the MetadataDescriptor is associated with the interface of the WS-Resource, and would not vary between different implementations of the interface. Information in the MetadataDescriptor provides clients of a WS-Resource the potential for greater understanding of the behavior of that WS-Resource.

The second purpose is to provide information about the value restrictions of the resource properties in the resource properties document for the WS-Resource. This additional information may be associated with implementations of the interface as well as with the WSDL interface definition.

The single portType that describes the manageability interface for a manageable resource type is derived from various other manageability portTypes. With WSDL 1.1, physically including, using copy and paste, the operations from each of these portTypes into the definition of the most
derived portType achieves this inheritance. Each of the portTypes from which a manageable resource’s portType is derived may also have a MetadataDescriptor to augment its description. Each of the portTypes MAY have an optional attribute information item that references a MetadataDescriptor component by its namespace qualified name (QName). The MetadataDescriptors also document the list of MetadataDescriptors they extend. A MetadataDescriptor may extend the information of another MetadataDescriptor by adding information about any new features in the new portType or describing other constraints associated with the more specialized or derived portType.

A WS-Resource can have a MetadataDescriptor associated with them through a resource property element in the resource property document. This document standardizes the form of the WS-Resource MetadataDescriptor that contains metadata information about a WS-Resource’s interface so that clients of that interface may reason about implementations of the interface at both design time and run time. The syntax of a preferred XML serialization of the information model is also described.


1.1 Goals and Requirements

The goal of this document is to define the terminology, concepts, information model and XML definitions needed to express the metadata requirements of Web services and stateful resources as defined by the implied resource pattern [State Paper].

1.1.1 Requirements

In meeting this goal, the specification MUST address the following specific requirements:

- Define an information model representing metadata about operations, resource properties and notification messages associated with a WS-Resource interface.
- Define a standard annotation for associating metadata descriptions with other description artifacts of the WS-Resource, particularly its WSDL 1.1 portType and its resource properties document definition.
- Define the standard schema for representing the aspects of the information model.

1.1.2 Non-Goals

The following topics are outside the scope of this specification:

- It is not an objective of this specification to define new message exchanges required to access the metadata from a WS-Resource.
- It is not an objective of this specification to describe the means required to store the metadata for a WS-Resource.

1.2 Terminology

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]. When describing abstract data models, this specification uses the notational convention used by the [XML Infoset]. Specifically, abstract property names always appear in square brackets (e.g., [some property]).
This specification uses a notational convention, referred to as “Pseudo-schemas” in a fashion similar to the WSDL 2.0 Part 1 specification [WSDL 2.0]. A Pseudo-schema uses a BNF-style convention to describe attributes and elements:

- `?' denotes optionality (i.e. zero or one occurrences),
- `*` denotes zero or more occurrences,
- `+` one or more occurrences,
- `['` and `']` are used to form groups,
- `|` represents choice.
- Attributes are conventionally assigned a value which corresponds to their type, as defined in the normative schema.

```
<-- sample pseudo-schema -->
<element
  required_attribute_of_type_QName="xs:QName"
  optional_attribute_of_type_string="xs:string"?
  required_eloment />
<optional_eloment >/?
<one_or_more_of_these_elements >/+
[<choice_1 >/ | <choice_2 >/ ]*
</element>
```

### 1.3 Namespaces

The following namespaces are used in this document:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>xs</td>
<td><a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a></td>
</tr>
<tr>
<td>wsdl</td>
<td><a href="http://schemas.xmlsoap.org/wsdil">http://schemas.xmlsoap.org/wsdil</a></td>
</tr>
<tr>
<td>wsrp</td>
<td><a href="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-01.xsd">http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-01.xsd</a></td>
</tr>
</tbody>
</table>
2 Terminology and Concepts

The following definitions outline the terminology and usage in this specification. This section gives only brief description of these terms.

**Metadata:**

- Data about data, or information about information. In practice, metadata comprises a structured set of descriptive elements to describe an information resource.

**MetadataDescriptor:**

- A unit of containment for specific metadata for various aspects of a WS-Resource’s interface, particularly those elements associated with the WS-Resource’s WSDL 1.1 portType.

**MetadataDescriptor Document:**

- An XML instance document whose root is a Definitions element from the wsrmd: namespace. This document contains definitions for zero or more MetadataDescriptor components.
3 Example

In the following example there are “Windows XP” WS-Resources whose values are projected from the implementation of the OperatingSystem portType.

3.1 The OperatingSystem portType

The OperatingSystem portType defines operations and a resource properties document that describes the Web services interface to operating system resource instances. As well as providing a mechanism for interacting with the operating system itself, this portType also describes properties, which represent the hardware of the underlying machine on which the operating system is running.

The OperatingSystem portType is derived from various other manageability portTypes – these illustrations use some of the function from the Identification manageability portType. As is required with WSDL 1.1, this derivation is achieved by physically including the definitions function from each of these portTypes in the definition of the OperatingSystem portType. The OperatingSystem portType also uses the DerivesFrom attribute extension to WSDL 1.1 portTypes proposed within WS Resource Properties [WS-ResourceProperties].

The OperatingSystem portType is sketched as follows:

```
(001) ... xmlns:os="http://example.com/ns/OperatingSystem"
(002) ... xmlns:id="http://example.com/ns/Identification"
(003) ...
(004) <portType name="OperatingSystem"
(005)  wsrf-rp:ResourceProperties="os:OSResourceProperties"
(006)  wsrf-rp:DerivesFrom="id:Identification"
(007)  wsrmd:metadataDescriptor="os:OperatingSystemMetadataDescriptor"
(008)  wsrmd:metadataDescriptorLocation="http://example.com/ns/OperatingSystem"
(009) http://example.com/metadataDescriptors/OperatingSystem.wsrmd" >
(010) ...
(011) </portType>
```

Line 004 contains a portType declaration for a portType named OperatingSystem in the namespace corresponding to the os: namespace prefix declaration.

Line 005 indicates the global XML element declaration of the root element of the resource properties document associated with any WS-Resource whose Web service implements the os:OperatingSystem portType.

Line 006 indicates this portType is derived from the portType named Identification from the namespace corresponding to the id: namespace prefix declaration. This means that the os:OperatingSystem portType includes all of the resource properties and operations defined by the id:Identification portType.

Line 007 identifies that a MetadataDescriptor has been defined for this interface identified by the QName os:OperatingSystemMetadataDescriptor.

Line 008 and 009 indicate that information about MetadataDescriptors in the namespace corresponding to the os: namespace prefix declaration can be found by dereferencing http://example.com/metadataDescriptors/OperatingSystem.wsrmd.
3.2 Operating System Properties

Following are a subset of the resource properties of the operating system. Following that are the MetadataDescriptors for the Identification portType, the Operating System portType and the Windows XP WS-Resource.

3.2.1 Operating System Resource Property definitions

```xml
(001) ...
(002) xmlns:os="..."
(003)
(004) <element name="OSResourceProperties">
(005) <complexType>
(006) <sequence>
(007) <element ref="id:ResourceType" minOccurs="0" maxOccurs="1"/>
(008) <element ref="id:ResourceID" minOccurs="0" maxOccurs="1"/>
(009) <element ref="os:numberOfProcesses" minOccurs="0" maxOccurs="1"/>
(010) <element ref="os:totalSwapSpaceSize" minOccurs="0" maxOccurs="1"/>
(011) <element ref="os:processor" minOccurs="1" maxOccurs="unbounded"/>
(012) </sequence>
(013) </complexType>
(014) </element>
(015)
(016) <element name="numberOfProcesses" type="xsd:int"/>
(017) <element name="totalSwapSpaceSize" type="xsd:unsignedLong"/>
(018) <element name="processor" type="os:Processor"/>
(019)
(020) <complexType name="Processor">
(021) <sequence>
(022) <element name="processorFamily" type="xsd:string" minOccurs="1" maxOccurs="1"/>
(023) <element name="manufacturer" type="xsd:string" minOccurs="1" maxOccurs="1"/>
(024) <element name="serialNumber" type="xsd:string" minOccurs="1" maxOccurs="1"/>
(025) </sequence>
(026) </complexType>
```

3.2.2 Identification Property definitions

```xml
(001) <element name="IdentificationResourceProperties">
(002) <complexType>
(003) <sequence>
(004) <element ref="ResourceType" minOccurs="0" maxOccurs="1"/>
(005) <element ref="ResourceID" minOccurs="0" maxOccurs="1"/>
(006) </sequence>
(007) </complexType>
(008) </element>
(009)
```
### 3.2.3 MetadataDescriptor for Identification portType

An example MetadataDescriptor document for the Identification portType is detailed below. For the purposes of this example this MetadataDescriptor document will be located at the URL http://example.com/metadataDescriptors/Identification.wsrmd.

```xml
<Definitions xmlns="http://docs.oasis-open.org/wsrf/2004/10/wsrf-WS-ResourceMetadataDescriptor-1.0-draft-01.xsd"
             xmlns:id="http://example.com/ns/Identification"
             targetNamespace="http://example.com/ns/Identification">

  <MetadataDescriptor name="IdentificationMetadataDescriptor" interface="id:Identification"
                      wsdlLocation="http://example.com/ns/Identification.wsdl">
    <Property path="id:ResourceID" mutability="constant"
              modifiability="read-only" />
    <Property path="id:ResourceType" mutability="constant"
              modifiability="read-only" />

  </MetadataDescriptor>

</Definitions>
```

Line (001) contains a Definitions element defining MetadataDescriptor elements for the target namespace identified in line (004).

There is one MetadataDescriptor element child of this Definitions element (lines (005)-(016)).

The name of the MetadataDescriptor is contained in line (006). This together with the namespace prefix declaration in line (003) corresponding to the targetNamespace of the Definitions element means the QName of the MetadataDescriptor is id:IdentificationMetadataDescriptor.

Line (007) identifies the QName of the portType (interface) to which this MetadataDescriptor is associated. The location of WSDL for the Identification portType is expressed in the wsdlLocation attribute in lines (008)-(009). This follows the pattern of the wsdli:wsdlLocation attribute defined in the WSDL 2.0 specification [WSDL 2.0].

Lines (010)-(015) show two Property elements containing further metadata information about resource properties defined in the resource properties document for the Identification portType.

Lines (010)-(012) contain the first Property element that references the QName of the id:ResourceId resource property. Line (011) indicates that the id:ResourceId resource property will always have a constant value. Line (012) states that the id:ResourceId resource property is read-only, meaning that it cannot be changed by a requestor using Web services message exchanges such as the SetResourceProperties operation as defined in WS-ResourceProperties.
The second Property element, in lines (013)-(015), references the QName of the id:ResourceType resource property in line (013). This resource property element has the same metadata attributes as id:ResourceIdentifier.

3.2.4 MetadataDescriptor for OperatingSystem portType

For the purposes of this example the MetadataDescriptor document for the OperatingSystem portType is located at the URL http://example.com/metadataDescriptors/OperatingSystem.wsrmd.

The contents of the MetadataDescriptor for the OperatingSystem portType appear as follows:
Lines (001) to (038) contain a Definitions element for the http://example.com/ns/OperatingSystem namespace.

Lines (006) to (037) contain the definition of the MetadataDescriptor with QName os:OperatingSystem MetadataDescriptor.

Line (008) indicates that this MetadataDescriptor corresponds to the OperatingSystem portType.

Lines (009)-(010) gives the location of the WSDL document that defines elements associated with the namespace URI associated with the os: prefix (i.e. the WSDL definitions element that defines the OperatingSystem portType).

Line (011) indicates that this MetadataDescriptor definition specializes a MetadataDescriptor definition named id:Identification. Lines (012) and (013) indicate that a MetadataDescriptor Definition can be found for the id: namespace at http://example.com/metadataDescriptors/Identification.wsrmd. Because this MetadataDescriptor component is a specialization of another MetadataDescriptor component, a consumer of this description must understand the combination of both MetadataDescriptor components.

Lines (014)-(040) contains the four properties described in this MetadataDescriptor example.

Lines (014)-(022) contains the Property element describing the id:ResourceType resource property from the Identification portType. This Property element augments any Property element for id:ResourceType that may appear in the MetadataDescriptor for the Identification MetadataDescriptor. Lines (015)-(020) contain an additional set of ValidValues that the id:ResourceType resource property may contain.

Lines (023)-(025) contain the os:numberOfProcesses Property element which references the QName of the os:numberOfProcesses resource property. Line (024) indicates that the value of the os:numberOfProcesses may change over time. Line (025) indicates that, the os:numberOfProcesses can not be changed by a requestor using Web services message exchanges such as the SetResourceProperties operation as defined in WS-ResourceProperties [WS-ResourceProperties].

The next two Property elements all reference the same resource property QName, the os:processor. Each of these property components however address different sub-elements within the resource property element as identified by the value of the Property’s path attribute information item. The path is a path expression that defines the node within the descendant axis of the resource property to which the description applies.

Lines (026)-(028) describe aspects of the os:processor that apply to the entire contents of the os:processor resource property. Lines (029)-(040) describe valid values for a subcomponent of os:processor (its os:processorFamily child element).

### 3.2.5 MetadataDescriptor for Windows XP WS-Resource

The following MetadataDescriptor document describes a “Windows XP” WS-Resource. Note that there is no portType defined for “Windows XP”; the generic OperatingSystem portType is sufficient to define manageability interface for Windows XP. However, metadata associated with
Windows XP can be more specialized, therefore the MetadataDescriptor for Windows XP specializes the OperatingSystem MetadataDescriptor.

```
(001) <Definitions
(002)   xmlns="http://docs.oasis-open.org/wsrf/2004/10/wsrf-WS-
(003)     ResourceMetadataDescriptor-1.0-draft-01.xsd"
(004)   xmlns:id="http://example.com/Identification"
(005)   xmlns:os="http://example.com/ns/OperatingSystem"
(006)   targetNamespace="http://example.com/ns/OperatingSystem/WinXP">
(007)   name="WindowsXPMetadataDescriptor"
(008)   interface="os:OperatingSystem"
(009)   wsdlLocation="http://example.com/ns/OperatingSystem
(010)     /OperatingSystem.wsdl"
(011)   specializes="os:OperatingSystemMetadataDescriptor"
(012)   metadataDescriptorLocation="http://example.com/ns/OperatingSystem
(013)     /metadataDescriptors/OperatingSystem.wsrdm"

http://example.com/metadataDescriptors/OperatingSystem.wsrdm >
(014)   <Property path="id:ResourceType">
(015)     <ValidValues>
(016)       <id:ResourceType>OperatingSystem/id:ResourceType>
(017)       <id:ResourceType>MicrosoftWindows_XP/id:ResourceType>
(018)     </ValidValues>
(019)   </Property>
(020)   <Property path="os:processor/os:processorFamily" >
(021)     <ValidValues>
(022)       <os:processorFamily>Pentium Family/os:processorFamily>
(023)       <os:processorFamily>AMD/os:processorFamily>
(024)     </ValidValues>
(025)   </Property>
(026) </MetadataDescriptor>
(027) </Definitions>
```

Line (008) indicates that this MetadataDescriptor corresponds to the OperatingSystem portType, and line (011) indicates that this MetadataDescriptor specializes the “base” MetadataDescriptor for the OperatingSystem MetadataDescriptor at the given URL. Because this MetadataDescriptor component is an extension of another MetadataDescriptor component, a consumer of this description must understand the combination of both MetadataDescriptor components. MetadataDescriptor encloses two Property elements. Lines (014)-(019) contain a description of the id:ResourceType resource property, further constraining the set of ValidValues for a Windows XP WS-Resource. Lines (020)-(025) constrain the os:processorFamily sub-element of the os:processor resource property by allowing fewer options on (specializing) the ValidValues.
4 Logical Model for Metadata

The following figure shows a logical model depicting the relationship between the various elements of metadata description and those elements the metadata describes.

![Diagram showing the logical model of WS-Resource Metadata]

In our model, the unit of metadata containment is referred to as a "MetadataDescriptor". A MetadataDescriptor is used to describe aspects of a WS-Resource’s interface, particularly those elements associated with the WS-Resource’s WSDL 1.1 portType.

A MetadataDescriptor may contain metadata describing/constraining information about operations associated with the WS-Resource’s portType (or WSDL 2.0 interface).

A MetadataDescriptor may contain metadata describing/constraining resource property elements as contained within the Resource Properties document type definition associated with the WS-Resource’s portType. As defined in the WS-Resource Properties specification, each resource property element is defined as an XML Schema global element, in some namespace.
5 Information Model for WS-Resource Metadata

This section describes the information model for metadata describing/constraining WS-Resources. This model is depicted as a set of components with properties, each aspect of a WS-Resource metadata component having its own set of properties. An XML Infoset representation for these components is also provided. How the XML Infoset representation of a given set of WS-Resource MetadataDescriptor components is constructed is outside the scope of this specification.

A UML diagram of the information model for WS-Resource metadata is shown in the following figure:

Figure 2 - Information Model for WS-Resource Metadata Descriptor

We describe the Definitions, MetadataDescriptor, Property, and Operation components in the following sections.
6 Definitions Component

The Definitions component is a container for a set of MetadataDescriptor components (see section 7). The Definitions component defines a targetNamespace which forms the {namespace} property of all components it contains.

The properties of a Definitions component are as follows:

- {targetNamespace} a namespace URI that applies as the {namespace} property to all [child] components.
- {metadataDescriptors} a set of zero or more MetadataDescriptor components.

A Definitions component MAY also contain additional "extension" components added using the extensibility mechanism defined by this specification (See section 11).

The following is an XML representation of the Definitions component:

```
<Definitions
    targetNamespace="xs:anyURI "
    {anyAttribute}* >
<documentation />?
<MetadataDescriptor / > *
{any}*
</Definitions>
```

The Definitions element information item has the following Infoset properties:

- A [local name] of "Definitions".
- one or more attribute information items amongst its [attributes] as follows:
  - A REQUIRED targetNamespace attribute information item
    - The value of this attribute information item contains a URI that defines the {namespace} property of all [child] components.
    - The type of the targetNamespace attribute information item is xs:anyURI.
  - Zero or more namespace qualified attribute information items. The [namespace name] of such attribute information items MUST NOT be "http://docs.oasis-open.org/wsrf/2004/10/wsrf-WS-ResourceMetadataDescriptor-1.0-draft-01.xsd".
- Zero or more element information items amongst its [children], in order as follows:
  - An OPTIONAL documentation element information item (See section 10).
  - Zero or more MetadataDescriptor element information items (See section 6.2).
  - Zero or more namespace-qualified element information items. The [namespace name] of such element information items MUST NOT be "http://docs.oasis-open.org/wsrf/2004/10/wsrf-WS-ResourceMetadataDescriptor-1.0-draft-01.xsd".
6.1 Mapping Definitions Component Properties to an XML Representation

The mapping between the properties of the Definitions component and the XML Representation of the Definitions element information item is described as follows.

<table>
<thead>
<tr>
<th>Property</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>{targetNamespace}</td>
<td>The actual value of the targetNamespace attribute information item..</td>
</tr>
<tr>
<td>{metadataDescriptors}</td>
<td>The MetadataDescriptor definitions corresponding to all the MetadataDescriptor element information items in the [children] of the Definitions element information item (if any).</td>
</tr>
</tbody>
</table>

6.2 MetadataDescriptor components within a Definitions component

All MetadataDescriptor components defined in a given namespace must appear as [children] of a Definitions component with {targetNamespace} value the same URI as that namespace. All MetadataDescriptor components must be uniquely named, implying that the {name} property of the MetadataDescriptor component must be unique amongst the {metadataDescriptors} of a Definitions component.
7 MetadataDescriptor Component

The MetadataDescriptor component is a container for a set of metadata descriptions and constraints on a WS-Resource. The MetadataDescriptor component contains additional information that describes or constrains various aspects of a WS-Resource. For example, it provides additional information about the interface of the WS-Resource relevant to the management of the resource. In particular, it allows tools and applications, such as management applications, to be able to reason in detail about the WS-Resource both at runtime and at development time when no instances of the WS-Resource are available.

A MetadataDescriptor component can optionally specialize one or more other MetadataDescriptor components. In such cases the MetadataDescriptor component is regarded as containing the properties, operations and notifications of the MetadataDescriptor component(s) it specializes (if any), along with any properties, operations and notifications it directly defines. The MetadataDescriptor MUST NOT specialize itself either directly or indirectly.

The properties of a MetadataDescriptor component are as follows:

- \{name\} a name of a MetadataDescriptor component.
- \{namespace\} a namespace URI of the MetadataDescriptor component.
- \{QName\} a combination of the \{name\} and \{namespace\} of the MetadataDescriptor component.
- \{interface\} a QName identifying a Web services interface definition with which this MetadataDescriptor is associated.
- \{wsdlLocation\} a set of URI pairs, each pair associating a namespace URI with a URL of a document containing a WSDL definition of that namespace. This is a similar mechanism to that used in WSDL 2.0.
- \{metadataDescriptorLocation\} a set of URI pairs, each pair associating a namespace URI with a URL of a document containing a MetadataDescriptor Definitions definition of that namespace.
- \{extended MetadataDescriptors\} a set of zero or more QNames of MetadataDescriptor components.
- \{properties\} A set of zero or more Property components
- \{operations\} A set of zero or more Operation components

A MetadataDescriptor component MAY also contain additional “extension” components added using the extensibility mechanism defined by this specification (See section 11).

The following is an XML representation of the MetadataDescriptor component:

```
<MetadataDescriptor
    name="xs:NCName"
    interface="xs:QName"
    specializes="list of xs:QName"?
    wsdlLocation="list of xs:anyUri"?
    metadataDescriptorLocation="list of xs:anyUri"?
    {anyAttribute}* >
  <documentation />?
  <Property /> *
</MetadataDescriptor>
```
The MetadataDescriptor element information item has the following Infoset properties:

- A [local name] of “MetadataDescriptor”.
- two or more attribute information items amongst its [attributes] as follows:
  - A REQUIRED name attribute information item
    - The value of this attribute information item contains the name of this MetadataDescriptor component.
    - The type of the name attribute information item is xs:NCName.
  - A REQUIRED interface attribute information item
    - The value of this attribute information item contains a QName of a WSDL 1.1 portType element or WSDL 2.0 interface element associated with this MetadataDescriptor component.
    - The type of the interface attribute information item is xs:QName.
  - An OPTIONAL specializes attribute information item
    - The value of this attribute information item contains a list of QNames.
      - Each QName in this list MUST resolve to a ( QName) of a MetadataDescriptor component.
    - The type of the specializes attribute information item is list of xs:QName.
  - An OPTIONAL wsdlLocation attribute information item
    - The value of this attribute information item contains a list of pairs of URIs; where the first URI of the pair, which MUST be an absolute URI as defined in [URI], indicates a WSDL namespace name, and, the second a hint as to the location of a WSDL document defining WSDL components for that namespace name. The second URI of a pair MAY be absolute or relative.
    - The type of the wsdlLocation attribute information item is list of xs:anyURI.
  - An OPTIONAL metadataDescriptorLocation attribute information item
    - The value of this attribute information item contains a list of pairs of URIs; where the first URI of the pair, which MUST be an absolute URI as defined in [URI], indicates a MetadataDescriptor namespace name, and the second a hint as to the location of a MetadataDescriptor document defining MetadataDescriptor components for that namespace name. The second URI of a pair MAY be absolute or relative.
    - The type of the metadataDescriptorLocation attribute information item is list of xs:anyURI.
Zero or more namespace qualified attribute information items. The [namespace name] of such attribute information items MUST NOT be "http://docs.oasis-open.org/wsrf/2004/10/wsrf-WS-ResourceMetadataDescriptor-1.0-draft-01.xsd".

- Zero or more element information items amongst its [children], in order as follows:
  - An OPTIONAL documentation element information item (See section 10).
  - Zero or more Operation element information items (See section 7.3)
  - Zero or more namespace-qualified element information items. The [namespace name] of such element information items MUST NOT be "http://docs.oasis-open.org/wsrf/2004/10/wsrf-WS-ResourceMetadataDescriptor-1.0-draft-01.xsd".

### 7.1 Mapping MetadataDescriptor Component Properties to an XML Representation

The mapping between the properties of the MetadataDescriptor component and the XML Representation of the MetadataDescriptor element information item is described as follows.

<table>
<thead>
<tr>
<th>Property</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>{name}</td>
<td>The actual value of the name attribute information item.</td>
</tr>
<tr>
<td>{namespace}</td>
<td>The value of the {targetNamespace} of the [parent] Definitions component.</td>
</tr>
<tr>
<td>{QName}</td>
<td>The combination of the {name} and {namespace} properties.</td>
</tr>
<tr>
<td>{interface}</td>
<td>The actual value of the interface attribute information item.</td>
</tr>
<tr>
<td>{extended MetadataDescriptors}</td>
<td>The set of MetadataDescriptor definitions resolved to by the values in the specializes attribute information item if any, plus the set of MetadataDescriptor definitions in the {extended MetadataDescriptors} property of those MetadataDescriptor components.</td>
</tr>
<tr>
<td>{wsdlLocation}</td>
<td>The actual value of the wsdlLocation attribute information item (if present).</td>
</tr>
<tr>
<td>{metadataDescriptorLocation}</td>
<td>The actual value of the metadataDescriptorLocation attribute information item (if present).</td>
</tr>
<tr>
<td>{properties}</td>
<td>The Property definitions corresponding to all the Property element information items in the [children] of the MetadataDescriptor element information item (if any) plus the set of Property definitions in the {properties} property of the MetadataDescriptor definitions in {extended MetadataDescriptors}, if any.</td>
</tr>
</tbody>
</table>
| {operations}                | The Operation definitions corresponding to all the Operation
element information items in the [children] of the MetadataDescriptor element information item (if any) plus the set of Operation definitions in the {operations} property of the MetadataDescriptor definitions in {extended MetadataDescriptors}, if any.

7.2 Properties component of a MetadataDescriptor

The {properties} of a MetadataDescriptor contains a set of Property components, defining additional metadata and constraints on resource property elements (and attributes) associated with a MetadataDescriptor. The definition of a Property component’s scope is contained in Section 8.6. When two Property components within a MetadataDescriptor’s {properties} have the same {path}, then they need to be combined according to the semantics defined in section 8.7. When one Property component within a MetadataDescriptor’s {properties} has a {path} that overlaps with the {path} of another Property component within that MetadataDescriptor’s {properties} then they must be “reconciled” according to the semantics defined in section 8.7.

7.3 Operations Component of a MetadataDescriptor

The {operations} of a MetadataDescriptor contain additional metadata associated with operations defined on the Web service portType or interface of a WS-Resource. In particular an Operation component indicates whether the corresponding Web service operation is idempotent. If two Operation components within a MetadataDescriptor’s {operations} refer to the same WSDL operation definition, the Operation components need to be collapsed according to rules defined in section 9.2.
8 Property Component

The Property component is a container for a set of metadata descriptions and constraints on a specific Resource Property element or attribute thereof.

The properties of a Property component are as follows:

- `{path}` an identifier of the XML element or attribute to which the Property component applies. This is composed of a path expression indicating that a resource property itself, or one if its descendant elements or attributes is the scope of the Property component.
- `{pathTarget}` the XML element node or attribute node identified by the `{path}`.
- `{pathElement}` if the `{pathTarget}` is an XML element, then `{pathElement}` is the `{pathTarget}`. If the `{pathTarget}` is an XML attribute, then the `{pathElement}` is the XML element on which the `{pathTarget}` is defined.
- `{modifiable}` an `xs:string` enumeration of “constant”, “appendable”, or “mutable”.
- `{mutability}` an `xs:string` enumeration of “read-only” or “read-write”.
- `{subscribability}` an `xs:boolean` indicating, if true, that the Resource Property element associated with the `{pathElement}` can be the target of a subscription.
- `{validValues}` zero or one `ValidValues` components or zero or one `ValidValueRange` components.
- `{staticValues}` zero or one `StaticValues` components.
- `{attributes}` zero or more `Attribute` components.

A Property component MAY also contain additional “extension” components added using the extensibility mechanism defined by this specification.

The following is an XML representation of the Property component:

```
<Property
    path="xs:QName [/ [xs:QName | xs:NCName ]]* [/ @xs:QName | xs:NCName ]?”
    modifiability="[constant|appendable|mutable]”?
    modifiability="[read-only|read-write]”?
    subscribability="xs:boolean”?
    {anyAttribute}* >
</Property>
```

The Property element information item has the following Infoset properties:
• A [local name] of "Property".
• A [namespace name] of "http://docs.oasis-open.org/wsrf/2004/10/wsrf-WS-
ResourceMetadataDescriptor-1.0-draft-01.xsd".
• one or more attribute information items amongst its [attributes] as follows:
  o A REQUIRED path attribute information item
    β The value of this attribute information item MUST contain a QName of
    the Resource Property (an XML Schema global element definition)
    contained within the Resource Properties document associated with the
    portType or interface identified by {interface}. The Resource Property
    element must conform to the requirements specified for Resource
    β The value of this attribute information item MAY be further augmented
    with an attribute identification sub-expression, that indicates an attribute
    on the (pathElement) that is the scope of this Property component.
    β The type of the path attribute information item is a path expression
    defined by the following BNF:
      (1) path ::= ResourceProperty Qualifier [AttributeIdentifier]
      (2) ResourceProperty ::= QName
          (vc: QName MUST correspond to a resource property
          declaration).
      (3) Qualifier ::= ["\" [Qname | NCName]]*
          (vc: the path must locate a node in the descendant axis of the
          ResourceProperty element)
      (4) AttributeIdentifier ::= / @ [Qname | NCName]
          (vc: the AttributeIdentifier must identify an attribute information
          element contained on the element identified by the
          ResourceProperty and Qualifier productions).
    β The value of this attribute information item can therefore be viewed as a
    sequence of nodes in a path, similar to the nodes in a file specifier in a
    hierarchical file system.
  o An OPTIONAL modifiability attribute information item
    β The value of this attribute information item indicates whether a requestor
    can modify the value of the {pathTarget}.
    β The type of the modifiability attribute information item is an xs:string
    restricted to the following enumeration:
      • "read-only" – The value of the {pathTarget} can not be changed
        by Web services message exchanges such as the
        SetResourceProperty message as defined in WS-
        ResourceProperties.
      • "read-write" – The value of the {pathTarget} MAY be changed by
        Web services message exchanges such as the
        SetResourceProperty message as defined in WS-
        ResourceProperties. Note - If the value of the modifiability
attribute information item is “read-write” then the value of the
mutability attribute information item MUST NOT be “constant”.

β If the modifiability attribute information item is not defined, the value of
the modifiability property is “unknown”.

○ An OPTIONAL modifiability attribute information item

β The value of this attribute information item expresses how the value of
the {pathTarget} can change over time.

β The type of the modifiability attribute information item is an xs:string
restricted to the following enumeration:

• “constant”
  The values of the {pathTarget} MUST NOT change
  after WS-Resource creation.

• “mutable”
  The values of the {pathTarget} MAY change at any
time during the lifetime of the WS-Resource. Existing
values MAY be removed and new values MAY be
added.

• “appendable”
  The values of the {pathTarget} MAY have new
values added during the lifetime of the WS-Resource.
Once added those values MUST NOT be removed.

β If the modifiability attribute information item is not defined, the value of the
mutability property is “unknown”.

○ An OPTIONAL subscribability attribute information item

β The value of this attribute information item expresses whether the
Resource Property element associated with the {pathElement} can be
the target of a subscription. The default value is “false”. Note: The
actual subscription semantics are dependent on whatever notification
mechanism, if any, (such as WS-BaseNotification [WS-BaseNotification])
is supported.

○ Zero or more namespace qualified attribute information items. The [namespace
name] of such attribute information items MUST NOT be “http://docs.oasis-
open.org/wsrf/2004/10/wsrf-WS-ResourceMetadataDescriptor-1.0-draft-01.xsd”.

• Zero or more element information items amongst its [children], in order as follows:
  ○ An OPTIONAL documentation element information item (See section 10).
  ○ An OPTIONAL element information item from among the following:
    o A ValidValues element information item (See section 8.2)
    o A ValidValueRange element information item (See section 8.4)
  ○ An OPTIONAL StaticValues element information item (See section 8.5)
  ○ Zero or more namespace-qualified element information items. The [namespace
name] of such element information items MUST NOT be “http://docs.oasis-
open.org/wsrf/2004/10/wsrf-WS-ResourceMetadataDescriptor-1.0-draft-01.xsd”.

wsrf-WS-ResourceMetadataDescriptor-1.0-draft-01.doc 10/21/2004
Copyright © OASIS Open 2004. All Rights Reserved. Page 25 of 53
### 8.1 Mapping Property Component Properties to an XML Representation

The mapping between the properties of the Property component and the XML Representation of the Property element information item is described as follows.

<table>
<thead>
<tr>
<th>Property</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>{path}</td>
<td>The actual value of the path attribute information item.</td>
</tr>
<tr>
<td>{modifiability}</td>
<td>The actual value of the modifiability attribute information item (if specified) or the value is unknown.</td>
</tr>
<tr>
<td>{mutability}</td>
<td>The actual value of the mutability attribute information item (if specified) or the value is unknown.</td>
</tr>
<tr>
<td>{subscribability}</td>
<td>The actual value of the subscribability attribute information item (if specified) or the value is “false”.</td>
</tr>
<tr>
<td>{validValues}</td>
<td>The validValues definition corresponding to the ValidValues element information item in the [children] of the Property element information item (if specified) or the ValidValueRange element information item in the [children] of the Property element information item (if specified), otherwise empty.</td>
</tr>
<tr>
<td>{staticValues}</td>
<td>The staticValues definition corresponding to the StaticValues element information item in the [children] property of the Property element information item (if any).</td>
</tr>
</tbody>
</table>

### 8.2 XML Schema value space and {validValues}

When creating a resource property (ie defining an XML Global Element), the XML Schema designer defines the semantic of the property and uses XML Schema to express the value space of the resource property (based on the semantics of the property) and all of its descendant element information items and attribute information items. This is a different concept from what is expressed by defining the {validValues}. When specifying {validValues} in a metadata description, one does not redefine the semantic of the {pathTarget} nor its value space. Specifying {validValues} expresses constraints on the value space that are appropriate for the specific use of the {pathTarget}. This distinction should guide designers in deciding whether to use XML Schema mechanisms or a metadata description to restrict value space of a {pathTarget}. The value space defined by {validValues} for a {pathTarget} by the MUST be contained within the value space as defined by the XML Schema definition of the {pathTarget}.

### 8.3 ValidValues

The purpose of the ValidValues component is to restrict the set of valid values that a [parent] Property component's {pathTarget} may contain. The complete set of valid values for a {pathTarget} is defined by the {validValues} of each Property component of a given {path}.
For ValidValues components that correspond to \{pathTarget\} which are elements, the ValidValues component contains a set of zero or more XML fragments whose QName and type correspond to the type defined by the \{pathTarget\}.  

For ValidValues components that correspond to \{pathTarget\} which are attributes, the ValidValues component contains a set of zero or more character data fragments whose contents reflect the type defined by the \{pathTarget\}.  

If the \{validValues\} of a Property component is not empty, and contains a ValidValues description, then any Web service that implements the portType or interface identified by \{interface\} MUST ensure that the value(s) of the \{pathTarget\} of the \[parent\] Property component MUST correspond to one of the values enumerated within the set of \{validValues\}.  

Note: because the child element information items of a ValidValues element information item are XML fragments, it is not required that these fragments be validated (processContents is "skip").  

The properties of a ValidValues component are as follows:  

- \{values\} zero or more XML fragments that correspond to the type of the \[parent\] Property component's \{pathTarget\}.  

The following is an XML representation of the ValidValues component:  

```xml
<ValidValues
   {anyAttribute}* >
   <documentation />?
   {any}*
</ValidValues>
```

The ValidValues element information item has the following Infoset properties:  

- A [local name] of "ValidValues".  
- zero or more attribute information items amongst its [attributes] as follows:  
  - Zero or more namespace qualified attribute information items. The [namespace name] of such attribute information items MUST NOT be "http://docs.oasis-open.org/wsrf/2004/10/wsrf-WS-ResourceMetadataDescriptor-1.0-draft-01.xsd".  
- Zero or more element information items amongst its [children], either:  
  - Zero or more namespace-qualified element information items. The [namespace name] of such element information items MUST NOT be "http://docs.oasis-open.org/wsrf/2004/10/wsrf-WS-ResourceMetadataDescriptor-1.0-draft-01.xsd".  
  - Each element information item MUST be an XML fragment that corresponds to the type of the XML element identified by the \[parent\] Property component's \{path\}  
  - Note, because these are XML fragments, it is not expected that a processor of a MetadataDescriptor document would need to validate these element information items (processContents = "skip").  
  - or  
  - Zero or more character information items.
8.4 ValidValueRange

The ValidValueRange component is an alternative mechanism to specify the set of ValidValues for the [parent] Property component’s {pathTarget}. Unlike the ValidValues component, which specifies an enumeration of values, the ValidValueRange restricts the set of valid values for a {pathTarget} by specifying a range of possible values. This mechanism can only be used when the {pathTarget} is an XML attribute, or an XML element of simpleType.

ValidValueRange defines an optional inclusive lower bound of the range and optional inclusive upper bound of the range. Both MAY be specified. At least one MUST be specified. The values of the lower bound and upper bound (if specified) MUST correspond to the value space definition of the {pathTarget}. If the {lowerBound} of this attribute information is NOT specified, its default value is defined by the lowest possible value defined for the value space of the {pathTarget} or “undefined”. Similarly the default value of {upperBound} is the largest value for the value space of the {pathTarget} or “undefined”.

If the {validValues} of a Property component is not empty and contains a ValidValueRange description, then any Web service that implements the portType or interface identified by {interface} MUST ensure that the value(s) of the resource property (or child element thereof) as identified by the {path} of the Property component MUST correspond to a value within the range specified by {validValues}.

The properties of a ValidValueRange component are as follows:

- {lowerBound} the (inclusive) lower bound of the value space defined by this component for the [parent] Property component’s {pathTarget}.
- {upperBound} the (inclusive) upper bound of the value space defined by this component for the [parent] Property component’s {pathTarget}.
- {range} a range of values, bounded by the values of {lowerBound} and {upperBound}. The values within {range} MUST be compliant with any value space constraints specified on the type definition of the [parent] Property component’s {pathTarget}.

The following is an XML representation of the ValidValues component:

```xml
<ValidValueRange>  
  <lowerBound/>  
  <upperBound/>  
  <documentation/>  
</ValidValueRange>
```

The ValidValueRange element information item has the following Infoset properties:

- A [local name] of “ValidValueRange”.
- one or more attribute information items amongst its [attributes] as follows:
  - one or more attribute information items amongst:
    - An OPTIONAL lowerBound attribute information item
      - The value of this attribute information item defines an inclusive lower bound on the range of valid values to apply to the [parent] Property component’s {pathTarget}.
```
8.4.1 Mapping ValidValueRange Component Properties to an XML Representation

The mapping between the properties of the ValidValueRange component and the XML Representation of the ValidValueRange element information item is described as follows.

<table>
<thead>
<tr>
<th>Property</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>(lowerBound)</td>
<td>The actual value of the lowerBound attribute information item, (if defined) or the lowest possible value in the value space defined for the [parent] Property components (pathTarget) or undefined.</td>
</tr>
<tr>
<td>(upperBound)</td>
<td>The actual value of the upperBound attribute information item, (if defined) or the largest possible value in the value space defined for the [parent] Property components (pathTarget) or undefined.</td>
</tr>
</tbody>
</table>

8.5 StaticValues

The purpose of the StaticValues component is to define the minimum set of values that a [parent] Property component’s (pathTarget) must contain. The complete set of static values for a (pathTarget) is defined as the set of all StaticValues components defined in the {staticValues} of each Property component of a given {path}. 
For StaticValues components that correspond to {pathTarget} which are elements, the
StaticValues component contains a set of zero or more XML fragments whose QName and type
correspond to the type defined by the {pathTarget}.

For StaticValues components that correspond to {pathTarget} which are attributes, the
StaticValues component contains a set of zero or more character data fragments whose contents
reflect the type defined by the {pathTarget}.

If the {staticValues} of a Property component is not empty, any Web service that implements the
portType or interface identified by {interface} MUST ensure that all the value(s) defined in
{staticValues} appear in the {pathTarget}.

The values contained in a StaticValues component MUST conform to the XML Schema definition
of the {pathTarget}. Note: because the child element information items of a StaticValues element
information item are XML fragments, it is not required that these fragments be validated
(processContents is “skip”).

The properties of a StaticValues component are as follows:

- {values} zero or more XML fragments that correspond to the type of the [parent] Property
  component’s {pathTarget}.
- If the {pathTarget} is an XML element, the number of XML fragments within {values}
  MUST NOT be greater than the maxOccurs facet of the {pathElement} of the [parent]
  Property component.
- If the {pathTarget} is an XML attribute, the number of child character information items is
  limited to one (if the {pathTarget} has non-list simpleType) or unlimited (if the
  {pathTarget} has a list simpleType).

The following is an XML representation of the StaticValues component:

```
<StaticValues
  {anyAttribute}*
  <documentation />?
</StaticValues>
```

The StaticValues element information item has the following Infoset properties:

- A [local name] of “StaticValues”.
  ResourceMetadataDescriptor-1.0-draft-01.xsd”.
- zero or more attribute information items amongst its [attributes] as follows:
  - Zero or more namespace qualified attribute information items. The [namespace
    name] of such attribute information items MUST NOT be “http://docs.oasis-
    open.org/wsrf/2004/10/wsrf-WS-ResourceMetadataDescriptor-1.0-draft-01.xsd”.
- Zero or more element information items amongst its [children], either:
  - Zero or more namespace-qualified element information items. The [namespace
    name] of such element information items MUST NOT be “http://docs.oasis-
    open.org/wsrf/2004/10/wsrf-WS-ResourceMetadataDescriptor-1.0-draft-01.xsd”.
  - Each element information item MUST be an XML fragment that
    corresponds to the type of the XML element identified by the [parent]
    Property component’s {path}
8.6 Path of a Property Component

Each Property component describes a single resource property element. However, a Property
MAY pertain specifically to a descendent of a resource property element or an attribute thereof.
The (pathTarget) of the Property component refers to the XML element or XML attribute to which
the metadata contained in the Property component applies.

Because (path) uses a path-based expression (similar to hierarchical file systems) it is important
to understand that multiple different Property components may pertain to the same (pathTarget).

To reason about the relationship of the (path) of a Property component, it is useful to decompose
the (path) property into a sequence of nodes, Node₀, Node₁, … Nodeₙ. Each node in the path is
separated by a '/' symbol. We use the term Aᵢ to represent the iᵗʰ node of the (path) of Property A.
We use the term Aₙ to denote the final or least significant node in the (path) of Property A. The
path length can also be used to describe the number of nodes in the (path) of Property A.

Two Property components, A and B have overlapping paths if A₁ = B₁. This is equivalent to
saying that Property A and Property B pertain to the same Resource Property, but either or both
of those components may pertain only to a descendent of the resource property or attribute
thereof. For example, if the (path) of Property A is “tns:a/b” and the (path) of Property B is “tns:a”,
we say that the (path) of Property A and the (path) of Property B overlap.

Two Property components, A and B have identical path if, for each value of i, Aᵢ = Bᵢ. This is
equivalent to saying the (path) of Property A is equivalent to the (path) of Property B.

Property component A has narrower path than Property B, if the path length of Property B is
less than the path length of Property A, and for each value of i, where i is less than the path
length of B, Aᵢ = Bᵢ. This is equivalent to saying that Property A is scoped to a more specific sub
part of the resource property. For example, if the (path) of Property A is “tns:a/b” and the (path) of
Property B is “tns:a”, we say that Property A has narrower scope, it describes or constrains only
the descendent nodes of the resource property “tns:a” rooted at the element “b”. Property B has a
“broader” (or less narrow) scope, in that it pertains to the entire resource property “tns:a”.

8.7 Reconciling Property Components with Overlapping Path

A MetadataDescriptor component may contain many Property components, some of which have
overlapping path. The Property components with overlapping path MUST be reconciled, to
ensure that the MetadataDescriptor has a coherent information model for the set of (pathTarget)s
it describes.

There are two cases of overlapping path to consider:

1. Two properties have identical path,
2. One Property has narrower path.

If two Properties with overlapping path cannot be reconciled (according to the algorithms that
follow), then the MetadataDescriptor component must be considered “incoherent” and consumers
of the MetadataDescriptor SHOULD ignore all Property descriptions whose (path) overlaps with
the irreconcilable (path). The producer of the MetadataDescriptor MUST NOT introduce changes that render the MetadataDescriptor information “incoherent”.

### 8.7.1 Reconciling Properties with Identical Path

Any two Property components, labeled A and B that have the same path MUST be reconciled by replacing Property A and Property B in the `properties` of the [parent] MetadataDescriptor with a new Property C as described by the following rules (note, Property A and Property B described in each of the rules below does not necessarily refer to the same property component across the rules):

1. The `{path}` of Property C is identical (by definition) to both Property A and B.

2. If the `{modifiability}` is defined for both Property A and Property B, and has different values, then Property A and Property B are not reconcilable. If the `{modifiability}` is defined for both Property A and Property B and has the same value, the `{modifiability}` of Property C is that value. If the `{modifiability}` is defined in one, but not both Property components then the value of `{modifiability}` of the combined Property C is set to the defined value. If the `{modifiability}` is not defined in either Property component then the value of `{modifiability}` of the combined Property C is also not defined.

3. If the `{mutability}` is defined for both Property A and Property B, and has different values, then Property A and Property B are not reconcilable. If the `{mutability}` is defined for both Property A and Property B and has the same value, the `{mutability}` of Property C is that value. If the `{mutability}` is defined in one but not both Property components then the value of `{mutability}` of the combined Property C is set to the defined value. If the `{mutability}` is not defined in either Property component then the value of `{mutability}` of the combined Property C is also not defined.

4. The `{validValues}` of Property C is formed as follows.

   - If the `{validValues}` of both Property A and Property B are identical, then the `{validValues}` of Property C is a copy of either the `{validValues}` from Property A or Property B.
   - If the `{validValues}` is empty in one Property component but non-empty in the other Property component, then the `{validValues}` for Property C is set to the non-empty `{validValues}`.
   - If the `{validValues}` of both Property A and Property B are non-empty, but not identical, then the `{validValues}` must be compared. There are two cases to consider:
     - a) The `{validValues}` of both Property A and Property B are defined by a ValidValueRange component, or the `{validValues}` of Property A and Property B are both defined by a ValidValues component:
       - If the `{validValues}` of Property A is a subset of the `{validValues}` of Property B, then the `{validValues}` of Property C is the subset (a copy of the `{validValues}` of Property A).
       - If the `{validValues}` of Property A is not a subset of the `{validValues}` of Property B, then Property A and Property B are not reconcilable.
     - b) The `{validValues}` of Property A are defined by a ValidValues component and the `{validValues}` of Property B are defined by a ValidValueRange component:
• If the \{validValues\} of Property A is contained within the range defined by the \{validValues\} of Property B, then the \{validValues\} of Property C is a copy of the \{validValues\} of Property A.

• If the \{validValues\} of Property A is not contained within the range defined by the \{validValues\} of Property B, then Property A and Property B are not reconcilable.

5. The \{staticValues\} of Property C is the set union of the \{staticValues\} from Property A and the \{staticValues\} from Property B. The resulting set union MUST abide by the schema cardinality restriction of the \{pathTarget\}.

6. Any namespace-qualified attribute information items or namespace-qualified element information items (extensions) that appear in Property A or Property B are mapped into Property C according to extension specific semantics.

8.7.2 Reconciling a Property with Narrower Path

Any two Property components, labeled A and B, such that Property A has “narrower” path must be reconciled by replacing Property A in the \{properties\} of the \{parent\} MetadataDescriptor with a new Property C as described by the following rules (note, Property A and Property B described in each of the rules below does not need to be consistent across the rules):

1. The value of \{path\} of Property C is the same as for Property A.

2. The values of \{modifiability\} and \{mutability\} of Property C are formed as described in section 8.7.1.

3. The \{validValues\} of Property C is formed as follows. If \{validValues\} are defined for Property B and \{validValues\} are also defined for Property A the two Property components are not reconcilable. Otherwise the \{validValues\} for Property C are the same as those for Property A.

4. The \{staticValues\} of Property C is formed as follows. If \{staticValues\} are defined for Property B and \{staticValues\} are also defined for Property A then the two Property components are not reconcilable. Otherwise the \{staticValues\} for Property C are the same as those for Property A.

5. Any namespace-qualified attribute information items or namespace-qualified element information items (extensions) that appear in Property A or Property B are mapped into Property C according to extension specific semantics.
9 Operation Component

The Operation component contains a metadata description for a specific wsdl operation element.

The properties of an Operation component are as follows:

- {name} the name of the wsdl operation to which the Operation applies. This must correspond to a name of an operation defined for the WSDL interface identified by the {interface} property of the Operation component.
- {inputMessage} a QName of the WSDL 1.1 Message which corresponds to the value of the message attribute information item in the operation element information item identified by {name}. This property may be used to disambiguate two operation element information items that share the same {name}, but have different input messages.
- {outputMessage} a QName of WSDL 1.1 Message corresponds to the value of the message attribute information item in the operation element information item identified by {name}. This property may be used to disambiguate two operation element information items that share the same {name}, but have different output messages.
- {interface} a QName of a WSDL 1.1 portType or WSDL 2.0 interface to which the Operation component is associated.
- {operation} the wsdl operation to which this Operation component applies. It is identified by the {interface} scoping the wsdl portType or interface and the {name} identifying the operation name within that wsdl portType or interface. Since WSDL 1.1 does not require operation children of a portType to have unique names, the {inputMessage} and {outputMessage} may also be used to identify the {operation}.
- {idempotent} an xs:boolean indicating, if true, that the wsdl operation corresponding to this Operation component is idempotent.

An Operation component MAY also contain additional "extension" components added using the extensibility mechanism defined by this specification.

The following is an XML representation of the Operation component:

```
<Operation
  name="xs:NCName"
  inputMessage="xs:QName" ?
  outputMessage="xs:QName" ?
  idempotent="xs:Boolean" ?
  {anyAttribute}* >
  <documentation>? 
  {any}
</Operation>
```

The Operation element information item has the following Infoset properties:

- A [local name] of "Operation".
- one or more attribute information items amongst its [attributes] as follows:
9.1 Mapping Operation Component Properties to an XML Representation

The mapping between the properties of the Operation component and the XML Representation of the Property element information item is described as follows.

<table>
<thead>
<tr>
<th>Property</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>{name}</td>
<td>The actual value of the name attribute information item.</td>
</tr>
<tr>
<td>{interface}</td>
<td>The actual value of the {interface} of the [parent] MetadataDescriptor.</td>
</tr>
<tr>
<td>{inputMessage}</td>
<td>The actual value of the inputMessage attribute information item (if present); or, the value of the message attribute information item contained within the [child] input element information item of the (operation) (if it exists), otherwise, nil.</td>
</tr>
</tbody>
</table>
The actual value of the outputMessage attribute information item (if present); or, the value of the message attribute information item contained within the [child] output element information item of the (operation) (if it exists), otherwise, nil.

The wsdl operation identified by the (name) corresponding to a wsdl operation defined in the wsdl portType or interface defined by (interface). The values of (inputMessage) and (outputMessage) may also be required to distinguish two (operation) components that have the same value of their name attribute information element.

The actual value of the idempotent attribute information item (if specified) or the value “false”.

9.2 Reconciling Operation Components referring to the same WSDL operation

If two Operation components, Operation A and Operation B within a MetadataDescriptor component’s (operations) has the same (operation), they must be reconciled to ensure that the MetadataDescriptor has a coherent information model for the operations it describes. If the Operation components are reconcilable, they are replaced with a new Operation component, Operation C, according to the following rules:

1. The (name), (interface), (inputMessage), (outputMessage) and (operation) are identical in Operation A and B, therefore Operation C has the same values for (name), (interface), (inputMessage), (outputMessage) and (operation) as found in either Operation A or B.

2. If the (idempotent) property of Operation A is the same as the (idempotent) property of Operation B, then Operation C takes has the same value of (idempotent) as found in either Operation A or B. If the (idempotent) property of Operation A is different than the (idempotent) property of Operation B, then the Operation components are not reconcilable.

3. Any namespace-qualified attribute information items or namespace-qualified element information items (extensions) that appear in Operation A or Operation B are mapped into Operation C according to extension specific semantics.
10 Documentation Component

The WS-Resource Metadata MetadataDescriptor specification uses the documentation element information item as a container for human readable and/or machine processable documentation in a fashion similar to that defined for WSDL 2.0 [WSDL 2.0]. The content of the element information item is "mixed" content as defined in XML Schema [XML Schema]. The documentation element information item may be contained by any element information item defined in this specification.

The following is an XML representation of the Documentation component:

```xml
<documentation {anyAttribute}* >
  {any} *
</documentation>
```

The documentation element information item contains the following:

- A [local name] of “documentation”.
- A [namespace name] of "http://docs.oasis-open.org/wsrf/2004/10/wsrf-WS-
  ResourceMetadataDescriptor-1.0-draft-01.xsd".
- Zero or more attribute information items.
- Zero or more child element information items amongst its [children].
- Zero or more character information items amongst its [children].

11 Extensibility of Components

Many of the components defined in this specification are extensible, meaning the components may be extended to include element information items and attribute information items other than those explicitly specified in the normative description of the component.

The purpose of this extensibility is to allow developers to include element information items and attribute information items particular to a given usage of WS-Resource Metadata Descriptors. The presence of these extended element information items and/or attribute information items augment the set of properties associated with a component and thereby may alter the semantics of the component. The designer of these extensibility components SHOULD use this component only for domain-specific extensions to the information model and SHOULD avoid altering the semantics of the component in ways that reduce interoperability and user understanding of the component. Extensions to the MetadataDescriptor component model SHOULD define rules for reconciliation.

12 Obtaining a MetadataDescriptor Document

There are two mechanisms that a requestor can use to obtain a WS-Resource MetadataDescriptor document:

1. A specific attribute extension to WSDL 1.1 portType definition

12.1 Extending WSDL 1.1 PortType

A WS-Resource MetadataDescriptor document is associated with a WSDL 1.1 portType definition using an extension of the WSDL 1.1 portType element information item. If any aspect of the
If the portType is associated with a MetadataDescriptor document, then the portType element MUST
be extended in the manner described below. This extension is described as follows:

```xml
<wsdl:definitions ...>
  <wsdl:portType ...
    wsrmd:metadataDescriptor="xs:QName"?
    wsrmd:metadataDescriptorLocation="list of pairs of xs:anyURI"?
  ...
</wsdl:portType>
```

This definition is further constrained as follows:

- `/wsdl:portType/@wsrmd:metadataDescriptor`
  - If this attribute appears on a WSDL 1.1 portType element its value MUST be a QName
    that corresponds to a MetadataDescriptor component. Further, the value of the
    MetadataDescriptor component contained in that document MUST have {interface} that
    matches the QName of the portType containing @wsrmd:metadataDescriptor. Any
    service that implements this portType MUST be associated with a MetadataDescriptor
    that is, or specializes the MetadataDescriptor identified by the value of this attribute.

- `/wsdl:portType/@wsrmd:metadataDescriptorLocation`
  - If this attribute appears on a WSDL 1.1 portType element its value MUST be a list of URI
    pairs. In each pair the first URI corresponds to a namespace URI of a
    MetadataDescriptor Definitions element. The second URI corresponds to a URL at which
    can be found more information about that MetadataDescriptor namespace, such as an
    XML document containing a MetadataDescriptor Definitions element as its root element.

### 12.2 Using Resource Property Elements to expose MetadataDescriptors

Some WS-Resources will have MetadataDescriptors that specialize the MetadataDescriptor
defined for its portType. In these cases, the requestor will need to determine if such further
specialization has occurred and retrieve the more specialized MetadataDescriptor. It is for this
purpose that we define a distinguished resource property, called wsrmd:MetaDataDescriptor.

If the designer wishes to expose the MetadataDescriptor of a WS-Resource as a resource
property (in order to assist the service requestor in inspecting a WS-Resource’s
MetadataDescriptor), the designer MUST include a reference to wsrmd:MetaDataDescriptor in
the XML Schema definition of the resource property document associated with the WS-
Resource’s portType. The form of this resource property element is:

```xml
  <xsd:element name="MetaDataDescriptor"
    type="wsrmd:MetaDataDescriptorRef"/>
```

The constraints on this element are as follows:

- `/wsrmd:MetaDataDescriptorRef/@name`
  - This component of the MetadataDescriptorRef contains the QName of a
    MetadataDescriptor. Its value MUST correspond to a MetadataDescriptor component
    that describes/constrains the WS-Resource. Further, the value of the MetadataDescriptor
    wsrf-WS-ResourceMetadataDescriptor-1.0-draft-01.doc 10/21/2004
Copyright © OASIS Open 2004. All Rights Reserved.
component MUST have {interface} that matches the QName of the portType of the WS-
Resource.

/wsrm:MetadataDescriptorRef/@metadataDescriptorLocation

If this attribute appears on a MetadataDescriptorRef its value MUST be a list of URI pairs.
In each pair the first URI corresponds to a namespace URI of a MetadataDescriptor
Definitions element. The second URI corresponds to a URL at which can be found more
information about that MetadataDescriptor namespace, such as an XML document
containing a MetadataDescriptor Definitions element as its root element.
13 References

13.1 Normative


13.2 Non-Normative

Appendix A. Acknowledgments

The following individuals were members of the committee during the development of this specification:

In addition, the following people made contributions to this specification:

Nick Butler (IBM), Donald F Ferguson (IBM), Randy George (IBM), Heather Kreger (IBM), Martin Nally (IBM), Mark Potts (HP),
Appendix B. XML Schema for WS-ResourceMetadataDescriptor

The XML types and elements used in this specification are defined in the following XML Schema:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<--
OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.

Copyright (C) OASIS Open (2004). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

-->
```
This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

```xml
</xsd:schema>
```

```xml
<!-- ------------------------- Utility Types -------------------------- -->
<xs:simpleType name="PairsOfURIType">
  <xs:list itemType="xsd:anyURI" />
</xs:simpleType>

<!-- ----------------- PortType Attribute Extensions ------------------> 
<xs:simpleType name="SpecializesType">
  <xs:list itemType="xsd:QName" />
</xs:simpleType>

<xs:attribute name="specializes" type="wsrm:SpecializesType" />
<xs:attribute name="metadataDescriptor" type="xsd:QName" />
<xs:attribute name="metadataDescriptorLocation" type="wsrm:PairsOfURIType" />

<!-- ------------------ Documentation Component ---------------------- -->
<xs:complexType name="DocumentationType" mixed="true">
  <xs:sequence>
    <xs:element name="DocumentationType" mixed="true" />
  </xs:sequence>
</xs:complexType>
```
<xsd:complexType name="DefinitionsType">
  <xsd:complexContent>
    <xsd:extension base="wsrmd:DocumentedType">
      <xsd:sequence>
        <xsd:element ref="wsrmd:MetadataDescriptor" minOccurs="0" maxOccurs="unbounded" />
        <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax" />
      </xsd:sequence>
      <xsd:attribute name="targetNamespace" type="xsd:anyURI" use="required" />
      <xsd:attribute name="#other" processContents="lax" />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Definitions" type="wsrmd:DefinitionsType">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="wsrmd:DocumentedType">
        <xsd:sequence>
          <xsd:element name="documentation" type="wsrmd:DocumentationType" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:complexType name="DocumentedType">
          <xsd:sequence>
            <xsd:element name="documentation" type="wsrmd:DocumentationType" minOccurs="0" maxOccurs="1" />
To form a QName, the name of any MetadataDescriptor must be unique within a Definitions element.

```xml
<xsd:documentation />
<xsd:annotation>
  <xsd:selector xpath="wsrmd:MetadataDescriptor" />
  <xsd:element xpath="@name" />
<xsd:element>
</xsd:annotation>
```

```xml
<MetadataDescriptor name="xsd:NCName" interface="xsd:QName" specializes="list of xsd:QName"?
wsdlLocation="list of xsd:anyUri"?
metadataDescriptorLocation="list of xsd:anyUri"
{anyAttribute}* >
<documentation />
<Property /> *
<Operation /> *
{any}*
</MetadataDescriptor>
```

```xml
<complexType name="MetadataDescriptorType">
  <complexContent>
    <extension base="wsrmd:DocumentedType">
      <sequence>
        <element ref="wsrmd:Property"
minOccurs="0" maxOccurs="unbounded" />
        <element ref="wsrmd:Operation"
minOccurs="0" maxOccurs="unbounded" />
      </sequence>
      <attribute name="name" type="xsd:NCName" use="required"/>
      <attribute name="interface" type="xsd:QName" use="required"/>
      <attribute name="specializes" type="wsrmd:SpecializesType" />
      <attribute name="wsdlLocation" type="wsrmd:PairsOfURIType" />
    </extension>
  </complexContent>
</complexType>
```
<xsd:attribute name="metadataDescriptorLocation"
type="wsrmd:PairsOfURIType"/>
<xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>
</xsd:complexContent>
</xsd:complexType>
<xsd:element name="MetadataDescriptor"
type="wsrmd:MetadataDescriptorType"/>

<!-- Property Component -->

<Property path="xsd:QName[/[xsd:QName | xsd:NCName]*/[@xs:QName | xsd:NCName]]?"
mutability="[constant | appendable | mutable]"?
modifiability="[read-only | read-write]"?
subscribability="xsd:boolean"?
(anyAttribute)*>
<documentation/>
[<ValidValues> {any}* </ValidValues> | <ValidValueRange lowerBound='xsd:simpleType'
upperBound='xsd:simpleType'>]<ValidValueRange> ] ?
<StaticValues> {any}* </StaticValues> ?
(any)*
</Property>

<!-- PropertyType -->

<xsd:complexType name="PropertyType">
<xsd:complexContent>
<xsd:extension base="wsrmd:DocumentedType">
<xsd:sequence>
<xsd:choice>
<xsd:element ref="wsrmd:ValidValues"
minOccurs="0" maxOccurs="1"/>
<xsd:element ref="wsrmd:ValidValueRange"
minOccurs="0" maxOccurs="1"/>
</xsd:choice>
<xsd:element ref="wsrmd:StaticValues"
minOccurs="0" maxOccurs="1"/>
</xsd:sequence>
</xsd:complexType>
</xsd:complexType>
</xsd:complexType>
</xsd:element>
</xsd:complexType>
</xsd:document>
type="wsrmd:PathType" use="required"/>
      <xsd:attribute name="mutability" type="wsrmd:MutabilityType" />
      <xsd:attribute name="modifiability" type="wsrmd:ModifiabilityType" />
      <xsd:attribute name="subscribability" type="xsd:boolean" default="false" />
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Property" type="wsrmd:PropertyType"/>

<xsd:simpleType name="PathType">
  <xsd:restriction base="xsd:token">
    <xsd:annotation>
      <xsd:documentation>
        The pattern language in XSD is hard to follow and harder to write.

        Since the pattern is almost impossible to figure out, here is a key:

        QName pattern is: \\
        \[i-[:]]\[c-[:]*:}\[i-[:]]\[c-[:]*]\\n        an optional prefix bit:

        \ie any initial character (except :) followed by zero or more
        chars (not :)
        followed by a :

        \ie any "initial character" (except :) followed by zero or more
        chars (not :)

        The path pattern is
        xsd:QName [\[ xsd:QName | xsd:NCName \]* [\@ xs:QName | xs:NCName ]\]?
      </xsd:documentation>
    </xsd:annotation>
    <xsd:pattern value="(((i-[:]][c-[:]*)?\[i-[:]][c-[:]*]*/\(((i-[:]][c-[:]*)?:\[i-[:]][c-[:]*])\((\[i-[:]][c-[:]*])*/\(\@\((\[i-[:]][c-[:]*)?:\[i-[:]][c-[:]*])\)\)\)?)?\[i-[:]][c-[:]*])\((\[i-[:]][c-[:]*])*/\(\@\((\[i-[:]][c-[:]*)?:\[i-[:]][c-[:]*])\)\)\)?" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="MutabilityType"/>
<xs:restriction base="xsd:string">
  <xs:enumeration value="constant" />
  <xs:enumeration value="appendable" />
  <xs:enumeration value="mutable" />
</xs:restriction>

</xs:simpleType>

<xs:simpleType name="ModifiabilityType">
  <xs:restriction base="xsd:string">
    <xs:enumeration value="read-only" />
    <xs:enumeration value="read-write" />
  </xs:restriction>
</xs:simpleType>

<!---------------- Valid Values Component ----------------->
<ValidValues {anyAttribute}*> ?
{any}*
</ValidValues>

<!---------------- Valid Range Component ----------------->
<ValidValueRange lowerBound="xs:anySimpleType" upperBound="xs:anySimpleType" {anyAttribute}*> ?
{any}*
</ValidValueRange>
<xsd:complexType name="ValidValueRangeType">
  <xsd:sequence>
    <xsd:element name="ValidValue" type="wsrmd:ValidValueType" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="OperationsType">
  <xsd:sequence>
    <xsd:element name="Operation" type="wsrmd:OperationType" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="StaticValuesType" mixed="true">
  <xsd:sequence>
    <xsd:element name="documentation" type="wsrmd:DocumentationType" minOccurs="0" maxOccurs="1" processContents="lax"/>
    <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
  </xsd:sequence>
  <xsd:attribute name="lowerBound" type="xsd:nonNegativeInteger"/>
  <xsd:attribute name="upperBound" type="xsd:nonNegativeInteger"/>
  <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>

<xsd:complexType name="StaticValuesType">
  <xsd:sequence>
    <xsd:element name="ValidValue" type="wsrmd:ValidValueType" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
  </xsd:sequence>
<xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>

<xsd:element name="StaticValues" type="wsrmd:StaticValuesType"/>

<xsd:complexType name="OperationsType">
  <xsd:sequence>
    <xsd:element name="Operation" type="wsrmd:OperationType" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="ValidValueRange" type="wsrmd:ValidValueRangeType"/>

<!-- Static Values Component -->
<StaticValues>{anyAttribute}*
  <documentation>?
  {any}*
</StaticValues>

<!-- Operation Component -->
<Operation name="xs:NCName" idempotent="xs:Boolean">{anyAttribute}*>
  <documentation>?
  {any}?
</Operation>
<xsd:element name="Operation" type="wsrmd:OperationsType" />
</xsd:complexType>

<xs:complexType name="MetadataDescriptorRefType">
  <xsd:complexContent>
    <xsd:extension base="wsrmd:DocumentedType">
      <xsd:sequence>
        <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax" />
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:NCName" use="required" />
      <xsd:attribute name="idempotent" type="xsd:boolean" default="false"/>
      <xsd:attribute namespace="##other" processContents="lax"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xs:element name="MetadataDescriptorRef" type="wsrmd:MetadataDescriptorRefType" />
</xsd:complexType>

<xs:complexType name="MetadataDescriptorRefType">
  <xsd:complexContent>
    <xsd:extension base="wsrmd:DocumentedType">
      <xsd:sequence>
        <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax" />
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:QName" use="required" />
      <xsd:attribute name="metadataDescriptorLocation" type="wsrmd:PairsOfURIType" />
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xs:element name="MetadataDescriptorRef" type="wsrmd:MetadataDescriptorRefType" />
</xsd:complexType>
## Appendix C. Revision History

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>By Whom</th>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td>wd-01</td>
<td>2004-10-07</td>
<td>Tom Maguire</td>
<td>Initial version created based on work in response to issue 10.</td>
</tr>
</tbody>
</table>
Appendix D. Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.

Copyright (C) OASIS Open (2004). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.