Web Service Manageability – Specification

(WS-Manageability)

Version 1.0
September 10, 2003

Authors

Mark Potts, Talking Blocks (mark.potts@talkingblocks.com)
Igor Sedukhin, Computer Associates (igor.seduhkin@ca.com)
Heather Kreger, IBM (kreger@us.ibm.com)
Ellen Stokes, IBM (stokese@us.ibm.com)

THE SPECIFICATION OR THE PERFORMANCE OR IMPLEMENTATION OF THE CONTENTS THEREOF.

You may remove these disclaimers from your modified versions of the Specification provided that you effectively disclaim all warranties and liabilities on behalf of all copyright holders in the copies of any such modified versions you distribute.

The names and trademarks of the Authors may NOT be used in any manner, including advertising or publicity pertaining to the Specification or its contents without specific, written prior permission. Title to copyright in the Specification will at all times remain with the Authors.

No other rights are granted by implication, estoppel or otherwise.

Status of this Document

This document is provided as a submission to the OASIS Web Services Distributed Management (WSDM) Technical Committee and to the Web service community in general.

Abstract

As Web services become pervasive and critical to business operations, the task of managing Web services and implementations of the Web services architecture will be imperative to the success of business operations. Web services manageability is defined as a set of capabilities for discovering the existence, availability, health, performance, and usage, as well as the control and configuration of a Web service within the Web services architecture. This implies that Web services can be managed using Web services technologies. The importance of a standardized management model for Web services and the promise of Web services as a management integration technology has driven industry leaders in Web services and management technologies and applications to form a technical committee in OASIS called the Web Services Distributed Management (WSDM) Technical Committee.

The WSDM TC is chartered to develop the specifications for defining how to use Web services to manage any IT resource – the "WSDM Management Using Web Services" specification. They are also charted to define the manageability model for a Web service as an IT resource and how to access that model using Web services – the "WSDM Management OF Web Services" specification. WS-Manageability defines the later, the manageability model for a Web service and how to access that model using Web services. WS-Manageability does not provide a "Management Using Web Services" specification. But it does provide insight, illustration, and input into how "Management Using Web Services" should be developed to support Web services and Grid services communities through a concrete use case: Managing Web services.

WS-Manageability

WS-Manageability is composed of three documents, a concepts document, a normative specification, and renderings listing. These documents are summarized below:

The WS-Manageability – Concepts document outlines the scope and definitions for the specification. It provides an overview of Web services architecture and implications for management of that architecture. The Concepts document further defines the role of the manager in the Web services architecture and provides practical information on manageability implementation patterns and discovery considerations.

The WS-Manageability - Specification begins by introducing the general concepts of a manageability model in terms of manageability topics, (identification, configuration, state, metrics, and relationships) and the aspects (properties, operations and events) used to define them. These abstract concepts apply to understanding and describing the manageability information and behavior of any IT resource, not just Web services. We use these concepts to organize our approach to Web services manageability.

The manageability model for Web services endpoint is defined as concrete models in UML using the topics and aspects concepts, without implying any particular implementation or locus of implementation. Appropriate manageability interfaces are defined based on the UML manageability models. While some parts of this model may be useful for modeling the manageability of any IT resource, this specification is focused exclusively on the requirements of Web service endpoints and does not propose a complete generic resource manageability model. The WSDM "Management Using Web Services" specification may incorporate these more generic models.

The WS-Manageability – Representation document provides the interface definitions based on the model as WSDL 1.1 and GWSDL renderings. These definitions are meant to show how the topics and aspects concepts along with concrete models can influence the development of consistent Web services interfaces for accessing the manageability information of Web services. The interfaces illustrate how the manageability model for Web services can be divided into aspects of topics that apply to all manageable resources and aspects of topics that apply only to the manageability of Web service endpoints. The interfaces that may apply to all manageable resources may be incorporated into the "WSDM Management Using Web Services" specification. The details of these WSDL interfaces specific to managing a Web service endpoint may change to remain consistent with the "WSDM Management Using Web Services" specification when it is available.

This specification depends on Web services as a distributed platform. There are several common functions which are required for management using Web services that are not specific to management and not yet available as standardized Web service technologies. Interim solutions have been proposed in this specification which will be replaced by suitable standards when they become available. It is not expected that the WSDM TC will continue specification of or standardization of the interim solutions.

This specification defines the manageability model for Web services, and then uses Web services as a 'worked example' of how manageability interfaces in general, as well as Web services manageability interfaces, may be rendered based on one approach to management using Web services practices.

Notational Conventions

The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” are to be interpreted as described in RFC-2119 [RFC 2119]. This is not yet done consistently. This specification uses namespace prefixes throughout; they are listed in Table 1. Note that the choice of any namespace prefix is arbitrary at this time and not semantically significant.

Prefix	Namespace
wsmr	urn:wsmr:common:relationships
wser	urn:wsmr:webservice:endpoint:relationships
wsdl	http://www.w3.org/2002/07/wsdl
soap	http://schemas.xmlsoap.org/wsdl/soap/
gwsdl	http://www.gridform.org/namespaces/2003/gridWSDLExtensions
ogsi	http://www.gridforum.org/namespaces/2003/OGSI
xsd	http://www.w3.org/2001/XMLSchema
wsp	http://schemas.xmlsoap.org/ws/2002/12/policy

Table 1: Some standard prefixes and namespaces commonly used in this document.

Table of Contents

1 Base Web Service Manageability Model

This specification defines the manageability interface for the Web services endpoint. Many of the concepts and techniques introduced apply equally to the manageability interface of any resource, but only those that apply to the management of Web services are defined. The general concepts are not identified separately from those specific to Web services. The separation of the concepts will be addressed based on the WSDM Management Using Web Services Specification when available. This model depends on a Web services platform for all non-management specific functionality. Where parts of this platform are required, but not yet standardized in the platform, solutions have been introduced to allow easy implementation until the appropriate standards become available.

Section 3 outlines an approach on rendering the manageability model in WSDL and GWSDL descriptions. The actual renderings are provided in the WS-Manageability – Representations document.

The interface development techniques used in these sections should be considered during the development of the Management Using Web services specification.

1.1 Manageability Topics

The definition of the base Web service manageability is driven from specific management concerns (exposed by the requirements for manageable Web services). For each of the concerns the subject is broken down into management capabilities by a *Topic*. A topic covers a functional capability that supports management of a particular problem or management domain. For example, the state management for a manageable resource is a management topic. Defining manageability within topics enables incremental and modular development and support of manageability capabilities. The functional capability of a topic is described using as combination of three aspects: properties, operations, and events.

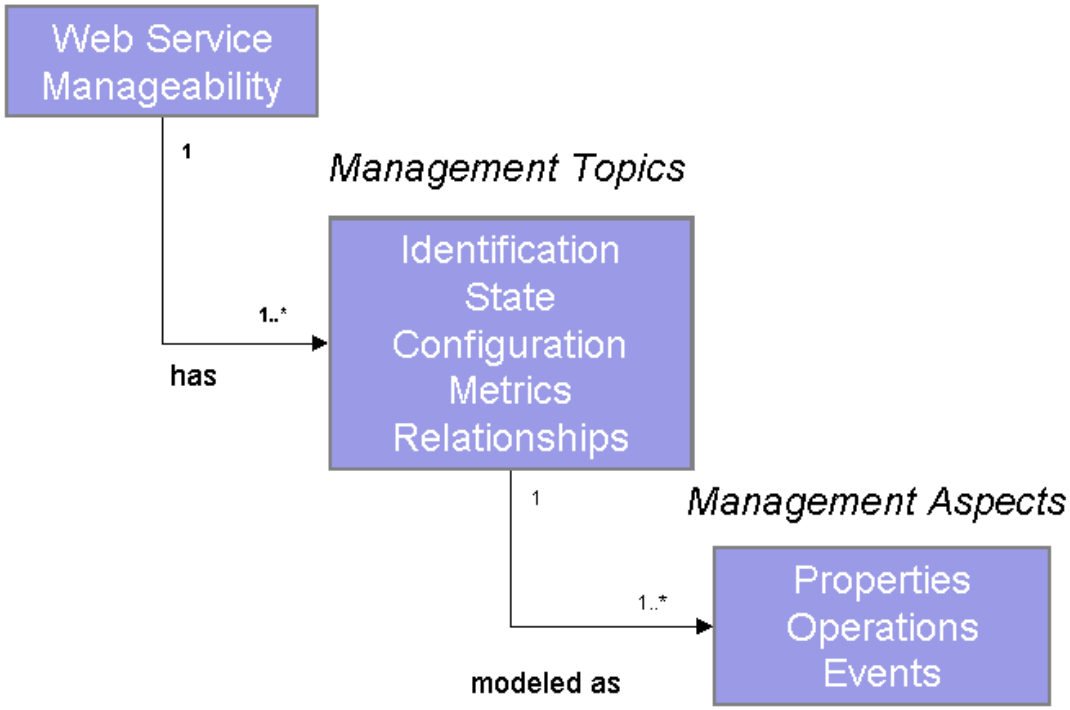


Figure 1: Web service Manageability

Topics and aspects are used to define a model with semantics for the management of Web services.

The topics covered by this specification are identification, state, configuration, metrics, and relationships.

1.1.1 Identification

The identification topic provides the functional capability to uniquely identify the resource being managed. This static information may include properties that are not required for unique identification, including descriptive and semantic information.

1.1.2 State

The State topic provides the functional capability to manage the actual operational state of a resource (i.e. up and down states in a lifecycle). The State topic defines properties about the operational state, operations to influence a change in state and events indicating when a state change has occurred.

1.1.3 Configuration

The Configuration topic provides the functional capability to manage the collection of properties whose values may influence the behavior of a resource. These

properties may be changed by the resource or may be changed by the manager which may cause behavior changes in the resource. Operations to access and change the configuration along with events indicating configuration changes are part of the configuration topic.

1.1.4 Metrics

The metrics topic provides the functional capability to manage metrics for a resource. The metric functional capability includes reset operations for metrics, and metric collection controls. Metrics are raw atomic, unambiguous, quantifiable information. The value of the metric captures the information at a point in time. Generally, these values are numeric, but may be strings as well.

Metrics can be contrasted with derived metrics that are calculated using a formula and metrics. For example, average response time during the last hour of execution is an example of a derived metric. Calculating such derived metrics is the responsibility of the manager.

1.1.5 Relationships

The relationships topic provides the capability to query associations that the resource (e.g., endpoint) participates in with other resources.

Relationships cover the associations that may exist between a resource and other resources of the same or disparate type. Relationships are important to management for problem isolation, root cause analysis, and impact analysis where resources are related in some way.

Relationship types that are defined between resources can have common or very specific semantics, requirements, and implications for management. For this reason it will be common to have topics that include operations that will create relationships that are viewed through this topic.

The semantics of the model are separated from any specific model rendition, such that the management systems, the services and the service environment all operate on the same formal understanding of the manageability. The information model is provided as a UML model which can be rendered in implementation specific techniques. For example, the information model can be then rendered into different representations of the model, for example DMTF's Common Information Model, a set of pure Web services interfaces, or a set of Grid services interfaces.

In this section the data model assumes the representation using the XML infoset (http://www.w3.org/TR/xml-infoset/) and refers to schema and type declaration and organization concepts defined in the XML Schema specification (http://www.w3.org/TR/xmlschema-0/). The model also uses URI, URN and URL schemes of addressing (http://www.w3.org/Addressing/) as well as QNames (http://www.w3.org/TR/xmlschema-2/#QName).

1.2 Aspects of Manageability

Capabilities grouped within a topic are defined as *Aspects* of the manageability, i.e. exposed properties, operations, and events required to understand and manage that topic. These aspects together are used to define interfaces for the topic. For example, the current state of a resource (e.g., an endpoint) is a property aspect of the state topic.

1.2.1 Properties

The property aspect provides a way to advertise or surface state information about a resource (e.g., endpoint).

Properties are expressed as named elements in an XML infoset. A property can be of a simple type or a complex type. Properties are manipulated directly through operations defined as part of the management interface (e.g. get/set operations) or by events that occur outside of the influence of the manager (e.g. a configuration file being updated).

Property change events may be emitted when a property's value is changed. Which property changes cause events to be emitted must be described. Emission of property change events may also be controllable by a manager to permit pausing and resuming the emission of property change events.

This specification uses the following patterns for all topics described by this aspect:

- Properties will be retrievable through strongly typed get operations. They may also be retrievable though a generic get function that can return any property for the resource.
- Properties can be specified as modifiable such that a manager may change their values. If a property is specified to be not modifiable, it may be constant and therefore contain a static value. An example of a constant property is the identification property whose value cannot be changed by a manager or will it change internally. Modifiable properties will have strongly typed set operations defined, while constant properties will not. Modifiable properties may also be changed using a generic set function that can change the value of any property for the resource.
- Resource specific properties can be added directly to the resource's manageability interface, to an extension of an existing topic, or to a new topic for different manageability purposes.
- Properties are defined in the context of a topic. Properties are classified into each topic by adding them to a list of properties that are specific to the topic. The property classification lists are also properties which are retrievable using strongly typed get operations and a generic get operation if it is available. This enables introspection of available properties for an interface.
- Advertising properties whose updates cause emission of property change events is done by adding the property's name to a list of properties which support change events

1.2.2 Operations

The Operation aspect captures methods to control the resource or to retrieve state information. Operations may cause temporary or permanent changes in the Web service's behavior.

This specification uses the following patterns for all topics described by this aspect:

- Operations are defined in the context of a manageability topic. The operations for a particular topic are specified by defining an exchange of messages for a purpose. The operation names should not conflict with property get and set operation naming conventions.
- Resource specific operations can be added directly to the resource's manageability interface, to an extension of an existing topic, or to a new topic for different manageability purposes.

1.2.3 Events

The Events aspect describes the indications of changes with respect to the resource. An event description is a set of information that describes the change. The information structure of an event description is defined by a named complex type. Notifications are messages containing the event descriptions that are transported to an interested party.

This specification uses the following patterns for all topics described by this aspect:

- The specific events for a given topic are described in each topic's section of this document, i.e. state change events are described in the state manageability topic for Web services endpoints.
- While notifications contain descriptions of specific event changes, some event description information is common across events. For example, all event descriptions contain time of the event, event source, event reporter, situation type, and generally a message text as well. A standard event format and XML schema is needed for management events to ensure alignment of semantics of the common event information. "The Canonical Situation Data and Common Base Event Specification 1.1", which has been submitted to the OASIS WSDM TC, defines a standard common event schema to be used for interoperability.
- This specification defines a pattern that allows a manager to poll for the "current events". The events about a Web service are made available to interested parties. Given that the existence of a standardized mechanism cannot be relied upon (see Note below), the default means to access events is a polling model where interested parties must poll the manageable Web service for its events. The set of historical events emitted to the manager are at the discretion of the manageable Web service. The events delivery mechanism in this specification will be replaced, as future enhancement, by a general-purpose standardized notification mechanism and interface when it is available..

***Note**: The definition of a standardised notification mechanism for subscription and publication of an event is outside of the scope of this work. Such a mechanism should be provided by the Web services environment (and preferably in a standardized interface) and used by the manageable services and managers. An example of a standard interface is the notification/subscription interfaces which are part of the Open Grid Services Infrastructure specification.*

The Topic and aspect definitions above define the base Web service manageability meta-model (Figure 2) which has been defined according to specific management concerns (indicated in the requirements for manageable Web services).

***Note**: The relationships model defined later in this work is specific to the manageability of Web service endpoints and is an extension of the Common Base Relationship (CBR) model (see Appendix A). The CBR model may be suitable for describing relationships between other IT resources, however that determination is not inferred by this work and is outside of the scope of this document. The anticiaption is that this work will adopt a standardised, genric representation for relationships when it becomes available.*

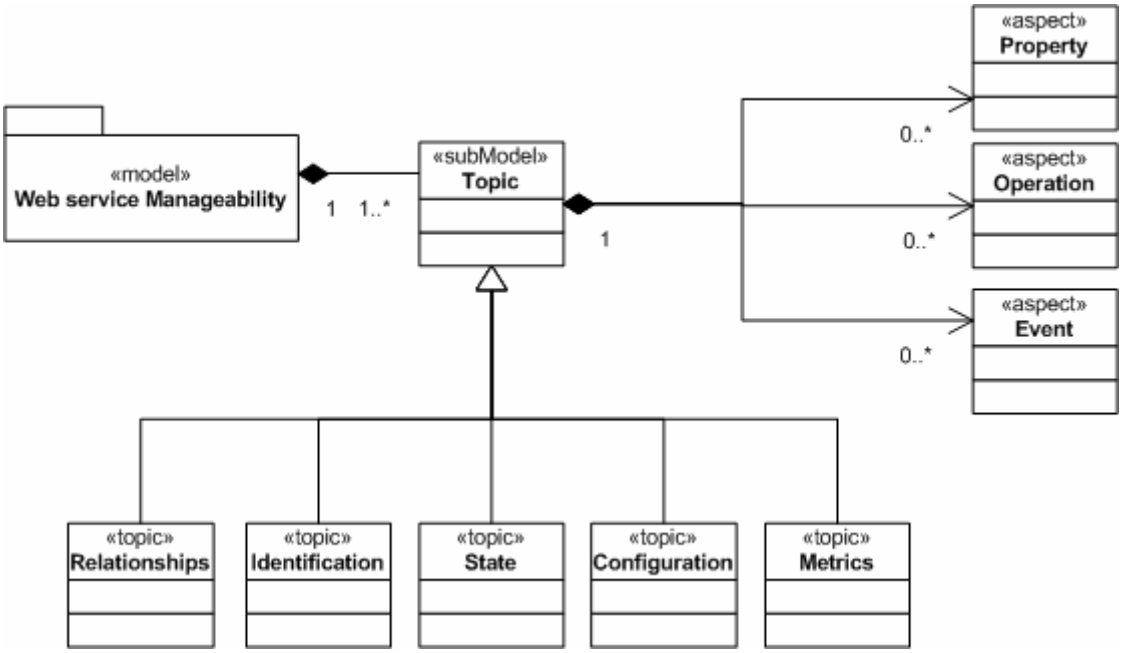


Figure 2: Base Web service Manageability meta-model

The next section defines the following manageability topics for a Web service endpoint: *Identification, State, Metrics, Configuration, Relationships*, and the management aspects (*Properties, Operations, and Events*) associated with each of the topics. The manageability topics and aspects defined for a Web service endpoint are based on, but not completely aligned with the W3C Web services Architecture Management Task Force Submission (http://lists.w3.org/Archives/Public/www-ws-arch/2003Mar/att-0001/W3c.Mtf.WSInstance.20030229.htm).

1.3 Manageability Topics of a Web Service Endpoint

This section defines manageability topics and their aspects for a manageable Web service endpoint. This section essentially defines an information model expressed in UML which is used to create a manageability interface for the Web service endpoint. The manageability

interface can be expressed using a variety of technologies, including WSDL 1.1, WSDL 1.2, and GWSDL.

This document uses the following conventions for the UML diagrams.

Data Types Legend

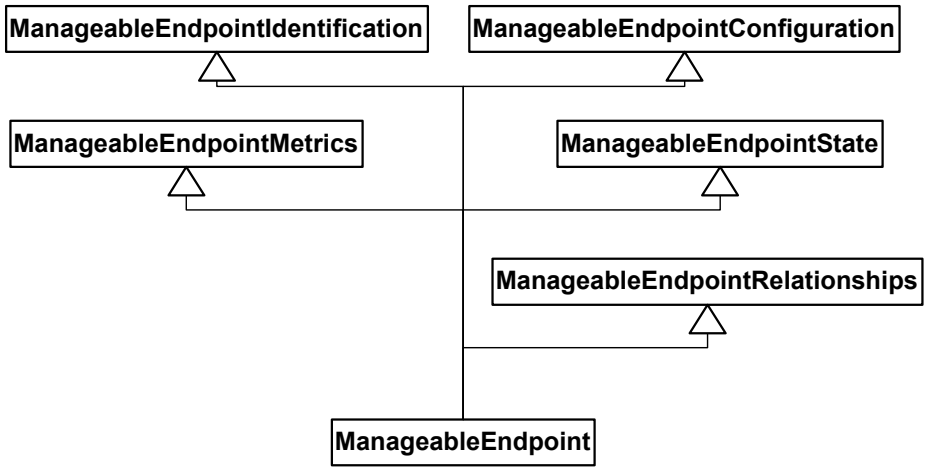
+ = regular property
- = nillable property

Information Model Legend

+ = modifiable property, or a callable method
- = not modifiable property (readonly)

The information model has a name defined in a namespace. Its name defines the type of a manageable element the model applies to. For a Web service endpoint, the manageability information model name is **ManageableEndpoint** which is defined in the **urn:wsdm:webservice:endpoint:manageability** namespace.

The manageability information model is an aggregate of sub-models each representing a manageability topic as shown in the following diagram.



1.3.1 Identification

The Identification topic for a Web service endpoint contains a set of immutable properties that uniquely identify the Web service endpoint that is being managed. There are no operations or events associated with the identification topic.

The following data types are used in this topic sub model. All data types specific to this sub model are defined in the **urn:wsdm:webservice:endpoint:identification** namespace.

EndpointIdentification
+identifier[1] : xsd:anyURI +name[1] : xsd:string +version[0..1] : xsd:string +semanticsIndentifier[0..1] : xsd:anyURI

- **identifier**
  The identifier of the managed Web service is the URI of its <port> element in a WSDL 1.1 document in the form consistent with that defined by the W3C WS Description WG. The current proposal from Arthur Ryman (http://lists.w3.org/Archives/Public/www-ws-desc/2002Dec/att-0021/01-URI-References.html) provides the following format.

```
<WSDL target namespace>#service(<serviceName>)/port(<portName>).
```

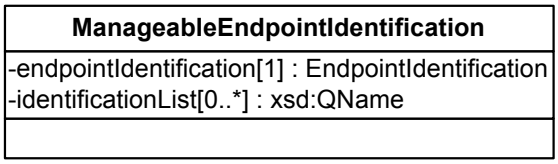
- **name**
  A human readable string name of the manageable service endpoint. It must match the value of the name attribute of the port element in the WSDL or GWSDL document.

- **version** [optional]
  The version of the managed Web service endpoint. The version should be expressed as a URI in a format consistent with that being defined by the WSDM TC.

- **semanticsIdentifier** [optional]
  A URI indicating the business semantics of the managed service offered via this endpoint. The format of semantic descriptions for a service is not currently standardized. The URI should be either a well understood keyword which indicates a set of well understood semantics for the service or it should be a URL which returns a document containing the semantics, preferably in a platform agnostic form like an XML or an HTML document.

The manageability information model for this topic has a name **ManageableEndpointIdentification** which is defined in the **urn:wsdm:webservice:endpoint:identification:manageability** namespace.

Following diagram shows this topic information model. The details are described in the following subsections.



1.3.1.1 Properties

- **endpointIdentification** [not modifiable] [constant]
  A set of identification properties for this Web service endpoint.

- **identificationList** [not modifiable] [constant]
  A set of QNames of the identification properties available for this Web service endpoint. Its initial list includes the endpointIdentification property element QName.

1.3.1.2 Operations

- **None defined**

1.3.1.3 Events

- **None defined**

1.3.2 State

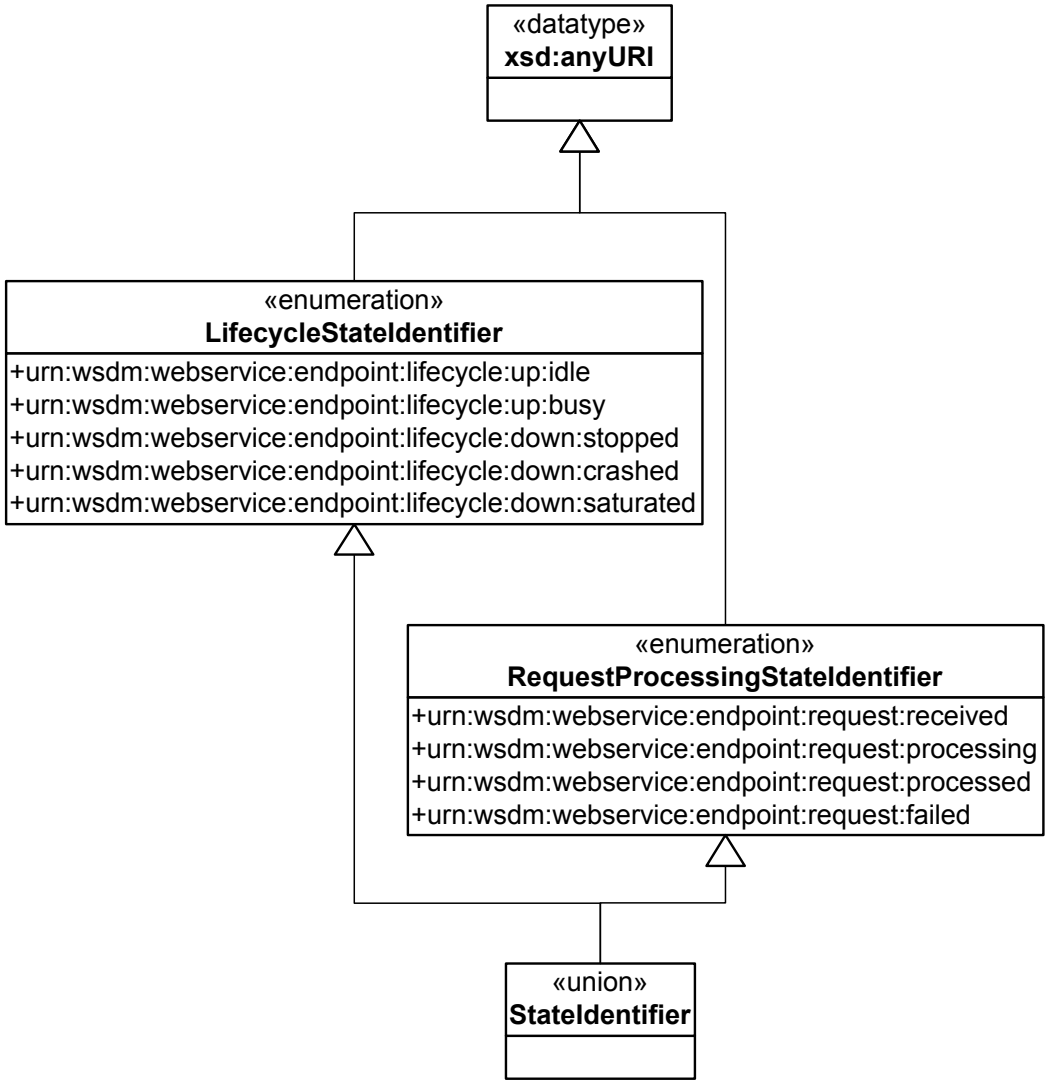
The operational, or lifecycle, state for a Web service is understood and managed through the State manageability model which supports obtaining the current state, changing the state, and events for state changes and events for request processing state changes.

The states and transitions are identified by URIs. The standard URIs for valid states and transitions are defined and are extensible.

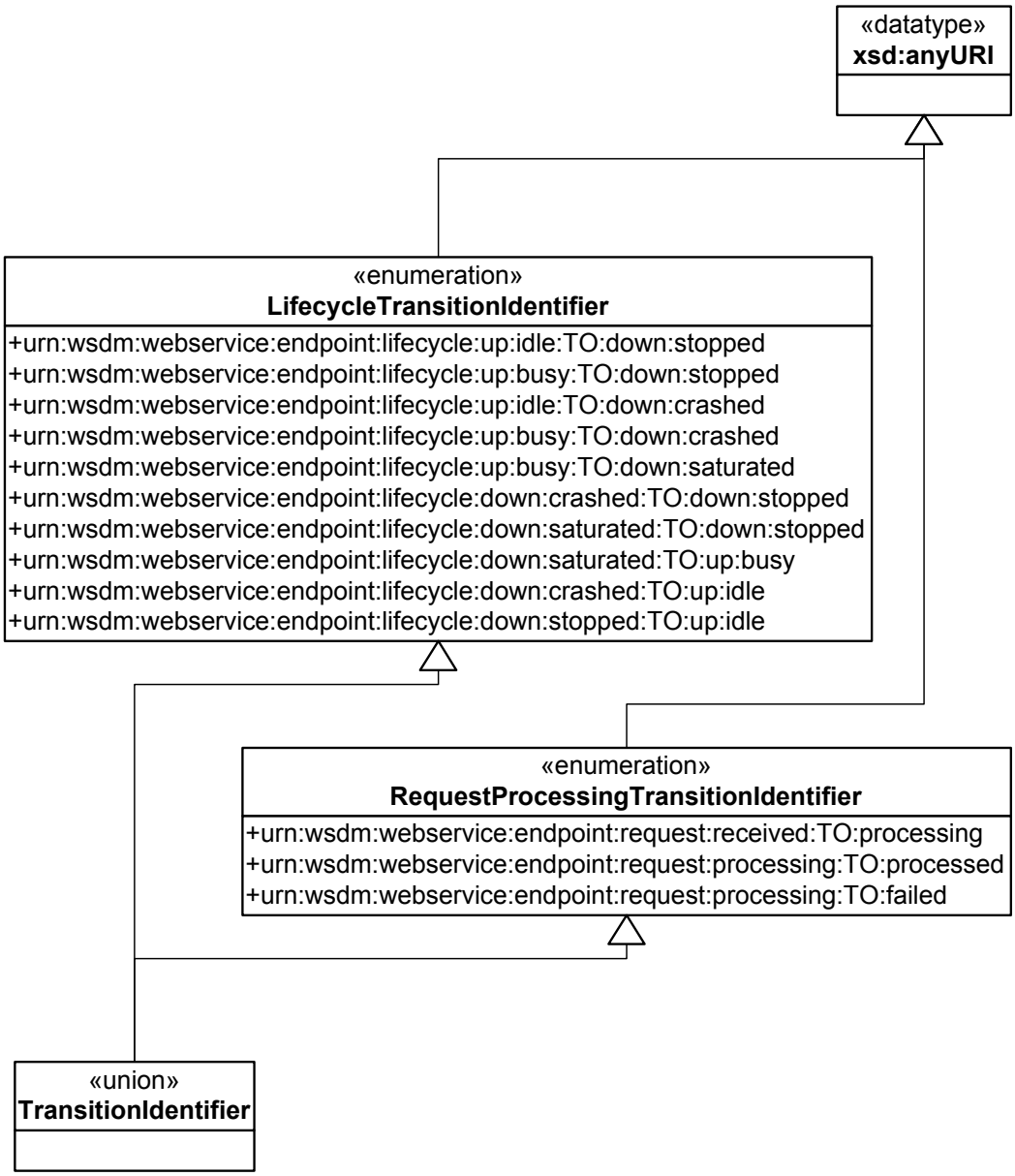
The state of a Web service endpoint is defined in the context of its runtime lifecycle and request processing activity as proposed by the W3C Web Services Architecture Management Task Force (http://www.w3.org/2002/ws/arch/2/11/W3C.MTF.ServiceLifecycle.20021111_clean.htm).

***Note**: The Management Task Force proposed the above to add clarity to certain aspects of the WSA and to make it possible to express sensible Web Services Management Architecture. At that time, the work was submitted to the W3C WSAG (11/11/2002), final consensus had not been reached within the MTF on the exact details of the proposed lifecycle and processing model. The lifecycle model is used in this document as the consensus opinion of the authors (see section 7.2).*

The following valid states are identified for a Web service endpoint.



The following valid state transitions are identified for a Web service endpoint.



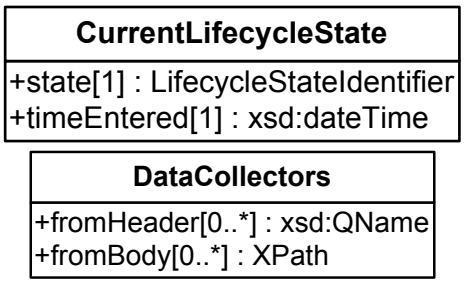
Transition identifiers are also used to identify the state change events.

The following data types are used in this topic sub-model. All data types specific to this sub-model are defined in the **urn:wsdm:webservice:endpoint:state** namespace.

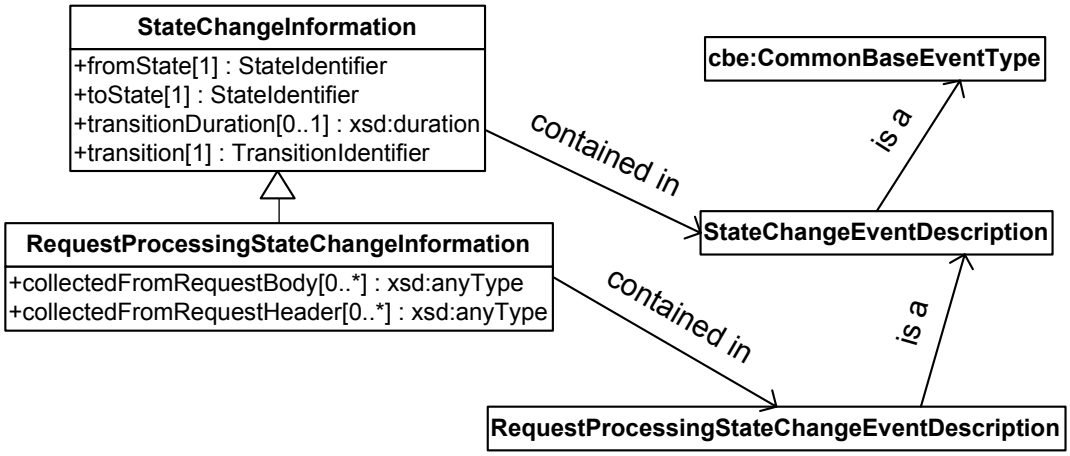
RequestProcessingTransitionSettings
+transition[1] : RequestProcessingTransitionIdentifier +dataCollectors[1] : DataCollectors

The request processing transition settings define the information to be collected from the request header by QName, and from the request body, by XPath expression (DataCollectors). This configuration is necessary for the Manager to be discrete about the set of information to collect. The collected information is passed with the request processing state change events.

The following data types are also used in this topic.



Following are event description data types for this topic.



The state change event description is a common base event message with an extended data field containing the state identifier, current state identifier, previous state identifier, a transition identifier and an optional duration of the transition.

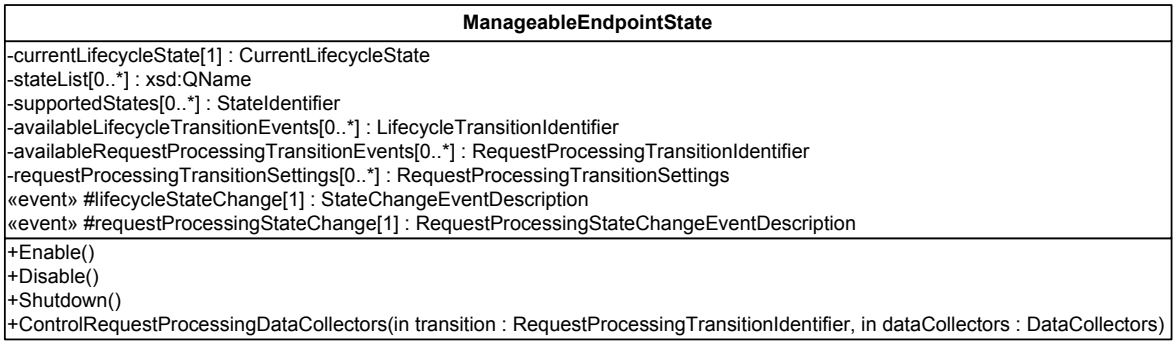
The request processing state change event description is also a common base event message with the same additional information as the state change event description. In addition the extended data field contains the XML information collected from the message header and XML information collected from the message body.

The processing state change event can be configured to include additional information from the request or reply messages. This includes anything in header, body, or attachment of the message. For example it may include operation identifier, parameter values, identity of requester as a WS-Security token, etc.

Managed resources may also support pausing and resuming the emission of events for a given transition. The Web services platform should provide a general purpose mechanism to control the emission of events by a manageable resource regardless of manageability topic the event is in. An interim solution to allow a manageable resource to control event emission has been provided as a 'common' interface until a standard specification that supports event emission control is available.

The manageability information model for this topic has a name **ManageableEndpointState** which is defined in the **urn:wsdm:webservice:endpoint:state:manageability** namespace.

Following diagram shows this topic's interface model. The details are described in the following subsections.



1.3.2.1 Properties

- **currentLifecycleState** [not modifiable] [not constant]
  Current lifecycle state reflects immediate situation of the Web service endpoint in its lifecycle. The value is a URI as defined by the list of valid states.

- **stateList** [not modifiable] [constant]
  A set of QNames of the state properties available for this Web service endpoint. Its initial list includes QNames of the following property elements currentLifecycleState, supportedStates, availableLifecycleTransitionEvents, availableRequestProcessingSettings, requestProcessingTransitionSettings properties.

- **supportedStates** [not modifiable] [constant]
  A list of valid state URIs relating to the lifecycle and request processing of the Web service endpoint. The manager is expected to have an understanding of this model. The understanding may be empirical (embedded in the manager by humans) or processed by the manager automatically from the model description documents (e.g. XML representation of the model) that are not defined here. The OGSI rendering

of the Web service endpoint model will include the definition of the lifecycle model in the GWSDL as defined by the "Common Manageability Model specification 1.0".

Note: *When the manageable service is 'plugged into' the larger context of an IT infrastructure or Grid which is manageable using Web services, it will be necessary to define the Lifecycle state model in XML as part of the manageability interface for the Web service*

- **availableLifecycleTransitionEvents** [not modifiable][not constant]
  This property is an array of URIs of each of the supported the lifecycle transitions for which events are supported. It allows a resource to advertise which events it supports emission for. If the URI of a lifecycle transition is in this list, then whenever that transition occurs an event will be emitted.

- **availableRequestProcessingTransitionEvents** [not modifiable][not constant]
  This property is an array of URIs of each of the supported the request processing transitions for which events are supported. It allows a resource to advertise which events it supports emission for. If the URI of a request processing transition is in this list, then whenever that transition occurs an event will be emitted.

- **requestProcessingTransitionSettings** [not modifiable][not constant]
  This property is an array of settings for each of the supported request processing transitions. It indicates the set of information that needs to be included in the description of the supported request processing state change events.

1.3.2.2 Operations

- **Enable**
  Changes the lifecycle state of a Web service endpoint from 'DOWN' to 'UP/IDLE'.

- **Disable**
  Changes the lifecycle state of a Web service endpoint from 'UP' or 'DOWN' to 'DOWN/STOPPED'.

- **Shutdown**
  Changes the state of all requests currently being processed by the service to 'FAILED' (cancels the requests) and disables the service (see Disable operation).

Note*: this does not imply any change in the status of the service implementation. It means, though, that any subsequent responses from the service implementation for cancelled requests will be lost.*

- **ControlRequestProcessingDataCollectors({<Request Processing Transition URI>|null}, {<Data Collectors>|null})**

Specifies what information has to be collected for a given request processing transition. If null is specified for a transition URI, this operation applies to all known request processing transitions. If null is specified for data collectors, this operation disables information collection for the given transition.

1.3.2.3 Events

State change events provide valuable information for management systems and can be used to calculate many derived metrics.

- **lifecycleStateChange**
  Service state change events occur whenever lifecycle state transitions complete and are represented by instances of Common Base Event descriptions.

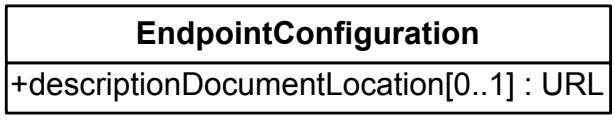
- **requestProcessingStateChange**
  Request processing state change events occur whenever request processing state transitions complete and are represented by instances of Common Base Event descriptions. The configuration property exists to allow a Manager to specify what additional information to include in the event.

***Note**: Manageable components are not required to use these events to calculate metrics or unconditionally disseminate them to management systems. In busy environments, these kinds of events can create a significant amount of overhead and implementations should consider their use judiciously. Events are not intended to be on all the time, they are intended to be 'turned on' (subscribed to, enabled) when there is an indication of problems. Configuration can identify which parts of the interaction context should be collected and returned.*

1.3.3 Configuration

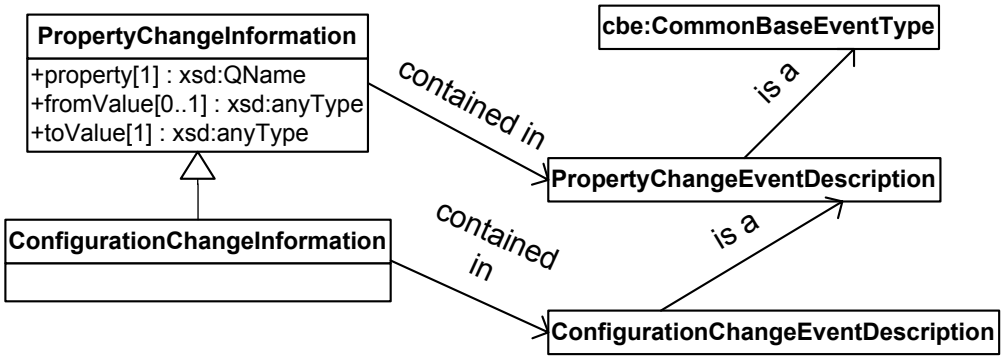
The configuration manageability of a Web service endpoint is described using the configuration topic which only identifies one configuration property, events indicating changes in a configuration property, and the ability to get and set the property. Configuration for a manageable endpoint, beyond the common configuration, should be defined by an extension to the manageability sub-model of the configuration topic.

The following data types are used in the configuration topic sub-model. All data types specific to this sub-model are defined in the **urn:wsdm:webservice:endpoint:configuration** namespace.



- **descriptionDocumentLocation** [optional, modifiable][not constant]
  The URL of the WSDL document associated with the managed endpoint.

Following are event description data types for this topic.



The configuration change event is defined for this topic. Its description is essentially the same as a generic property change event description which contains the common base event format with the identifier of the property, original value and current value in the extended data field. The identifier of the configuration property is the QName of the element representing the changed property (the QName of the XML element or a service data element in GWSDL that defines the property).

PropertyChangeInformation is generic, common and not specific to configuration of a manageable endpoint. These data types can be used in other manageability topics as well.

The manageability information model for this topic has a name **ManageableEndpointConfiguration** which is defined in the **urn:wsdm:webservice:endpoint:configuration:manageability** namespace.

The following diagram shows this topic's interface model. The details are described in the following subsections.

ManageableEndpointConfiguration
+endpointConfiguration[1] : EndpointConfiguration -configurationList[0..*] : xsd:QName -availableConfigurationEvents[0..*] : xsd:QName «event» #configurationChange[1] : ConfigurationChangeEventDescription

1.3.3.1 Properties

- **endpointConfiguration** [modifiable][not constant]
  A set of Web service endpoint configuration properties.

- **configurationList** [not modifiable][constant]
  This property contains a list of QNames of the properties that are configuration properties.

- **availableConfigurationEvents** [not modifiable][constant]

This property is a list of configuration properties for which events are available. This list is a subset of the configurationList. If a property's QName is in this list, then a change in that property's value will cause a configurationChange event to be emitted.

1.3.3.2 Operations

- **None defined**

1.3.3.3 Events

- **configurationChange**
  This event occurs whenever a configuration property value is changed. For this to happen, events must be enabled for the property.

1.3.4 Metrics

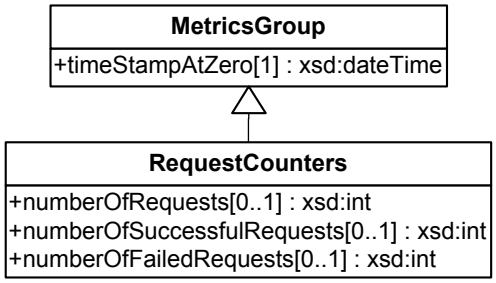
Metrics that Web services endpoints support are described in the metrics topic description along with event behavior for metric changes and operations to reset and control collection of values.

All data types specific to this sub-model are defined in the **urn:wsdm:webservice:endpoint:metrics** namespace.

Metrics are defined in groups of related metrics that can only be collected together. The metrics in a metric group are related to, dependent on, or provide semantics for other metrics in the group. Each metrics group is represented with a named element in the XML Schema. The QName of the element is the identifier of the group. A metrics group can contain other metrics groups. The wrapping group QName also refers to the combination of its component groups.

Every metric is defined as an element (of a complex or simple type). A QName of the element identifies the metric. Metrics in a group can be accessed individually or as part of the group.

The following metrics group is defined for a Web service endpoint.



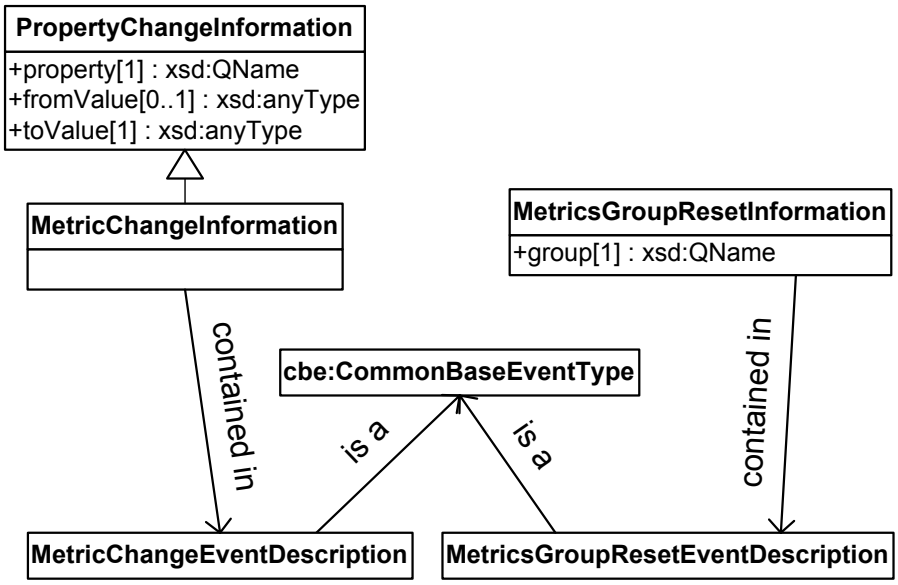
timeStampAtZero – the timestamp at which all the counters in the group were set or reset to 0.

numberOfRequests – the number of requests received (transitioned into request processing state urn:wsdm:webservice:endpoint:request:received:TO:processing) by a Web services endpoint since the timestamp in the timeStampAtZero field.

numberOfSuccessfulRequests – the number of requests received by a Web services endpoint since the timestamp in the timeStampAtZero field.that returned a successful response (transitioned into the request processing urn:wsdm:webservice:endpoint:request:processing:TO:processed state)

numberOfFailedRequests – the number of requests received by a Web services endpoint since the timestamp in the timeStampAtZero that returned a failed response (transitioned into the request processing urn:wsdm:webservice:endpoint:request:processing:TO:failed state)

Following are event description data types for this topic.

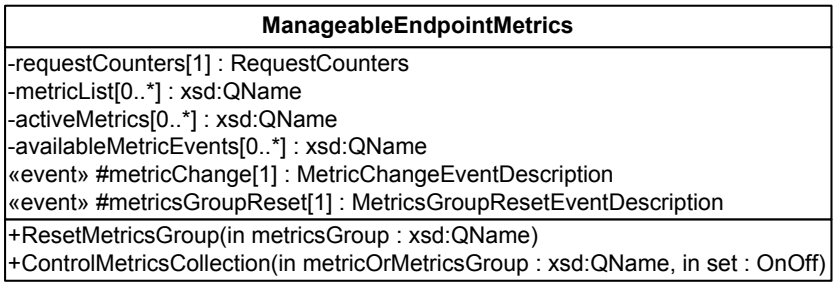


The metric change event description is a common base event with the metric property identifier (QName), original value, and current value in the extended data field. The event description may be extended for custom metrics.

The MetricsGroupResetEventDescription is emitted when the metric group is reset. It is a common base event information with the QName of the affected metrics group identifier in the extended data field.

The manageability information model for this topic has a name **ManageableEndpointMetrics** which is defined in the **urn:wsdm:webservice:endpoint:metrics:manageability** namespace.

Following diagram shows this topic interface model. The details are described in the following subsections.



1.3.4.1 Properties

- **requestCounters** [not modifiable][not constant]
  Contains values of the individual metrics in the Request Counter's metrics group and the timestamp at which time the counters were set at 0 or reset to 0. Since request counters metrics are interdependent they are all set to 0 at once.

- **metricList** [not modifiable][constant]
  A list of QNames of properties that are individual metrics or metric groups.

- **activeMetrics** [modifiable][not constant]
  A list of QNames of metrics or metric groups that are currently being collected and have values. The QNames in this list are a subset of the QNames in metricList. If any particular metric is not collected, its value is not available. An exception will be issued if an uncollected metric is polled.

- **availableMetricEvents** [not modifiable,][not constant]
  A list of metrics and metrics groups for which change events are available from the resource. If a metric's or metric group's QName is in this list then a metricChange event will be emitted whenever there is a change in that metric's or metric group's value. Any change in any value in a metric group will cause a metricChange event.

***Note**: You only need MetricsGroup.timeStampAtZero if lifecycle events are not available to the manager, such that it can calculate requests over time periods. This applies to the ResetMetricsGroup operation as well.*

***Note**: Many more metrics can be calculated based on this core set of metrics and information available through state transition events.*

1.3.4.2 Operations

- **ResetMetricsGroup({<metrics group QName>|null})**

Resets the value of all the metrics in the group. Sets all numeric values to 0 and remembers the time this happened in the timeStampAtZero field of the metric group. If **null** is specified, operation applies to all available metrics groups.

- **ControlMetricsCollection ({<metric or metrics group QName>|null},{on|off})**
  Enables or disables collection of values for a given metric or a given metrics group. If **null** is specified, operation applies to all available metrics and metrics groups.

1.3.4.3 Events

- **metricChange**
  This event occurs whenever the value of a metric is changed.

- **metricsGroupReset**
  This event occurs whenever a metric or the metrics in the group are reset.

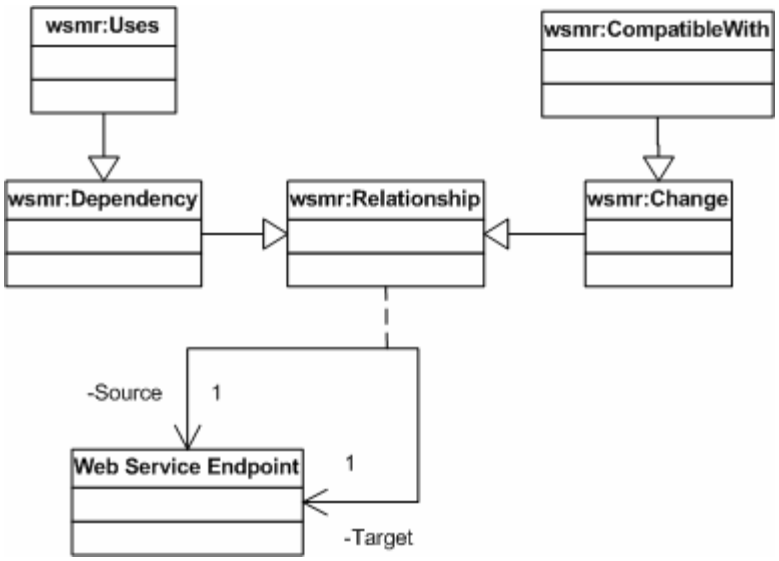
1.3.5 Relationships

Web service relationships are declared, between Web service endpoints in two ways, at an **interface** level, or an **endpoint** level. Interface level relationships can be defined declaratively in the WSDL definition of the Web service endpoint or made available through the manageability information model of a **ManageableEndpoint**, specifically the **ManageableEndpointRelationships** topic sub-model. Endpoint relationships are only made available by the **ManageableEndpointRelationships** interface.

The only differences in the Web service specific model from the Common Base Relationship Model (see Appendix A) is the extension of a **ManageableRelationships** to **ManageableEndpointRelationships** in the **urn:wsmr:webservice:endpoint:relationships:manageability** namespace and the definitions of specific relationships types for Web service endpoints.

1.3.5.1 Types of Web service endpoint relationships

Based on the current requirements for management of Web services, two specific types of relationships are identified. The specific relationships types aim to address the major concerns in managing Web service endpoints, Usage **Dependency** and **Compatibility** between Web services.



The generic Relationship type also allows for custom relationships and extensibility of the relationship model for relationships to other resources or infrastructure related to a Web services.

Relationships at the interface level allow for visibility into relationships even where the specific Web service endpoint is unknown. For example, one Web service "uses" another Web service (a dependency on the interface) but the specific endpoint used in each interaction is resolved dynamically at runtime. Another example is when compatibility declarations are made between interfaces.

The semantics of a relationship are defined by its type. The types of relationships identified based on manageability requirements surrounding Dependency and Change management are **Dependency:Uses** and **Change:CompatibleWith** (see 2.6).

- **urn:wsmr:webservice:endpoint:dependency:uses**
  Many Web service endpoints will rely on other Web service endpoints to provide their defined functionality. Consider the case of a service that "uses" other Web services to be available and coordinates interactions with those services (e.g. BPEL process exposed as a Web service). In such cases, it is important to address manageability on two fronts;
  1) **impact analysis** (e.g. understanding impact on *dependant* Web services caused by state changes that occur in an *antecedent* Web service) and
  2) **problem isolation and root cause analysis** (e.g. identifying the *antecedent* Web service causing problems with *dependant* Web services).

  A **uses** dependency can be bi-directional where Web service endpoints have defined roles within the relationship and are defined as one-to-many relationships in each direction between the source and target(s).

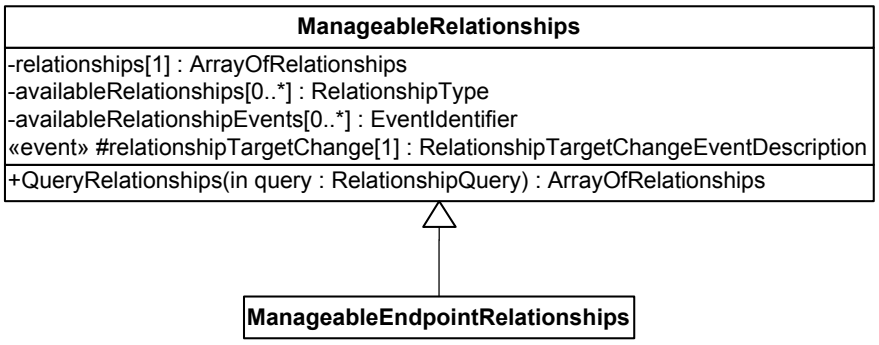
- **urn:wsmr:webservice:endpoint:change:compatibleWith**
  A "change" link defines relationships where a change has occurred, for example a re-deployment or an upgrade to an existing Web service. The specific change relationship identified for managing Web service endpoints is **compatibleWith**. Compatibility represents interface and semantic compatibility between Web services. It means that the same messages can be sent to either Web service endpoint (accomplishing the same task).

  A **compatibleWith** relationship, given its lineage, will be dependant upon a Web services versioning mechanism that we expect to be defined as part of WSDL 1.2 or the WSDM change description initiatives. The relationship can be bi-directional, but in many cases will be defined explicitly in one direction, e.g. "replaces" where deprecation of previous versions may be intended. In either case, the relationship is a one-to-many.

***Note**: The model defined here of relationships is specific to the manageability of Web service endpoints and is an extension of the Common Base Relationship (CBR) model (see Appendix A). The CBR model may be suitable for describing relationships between other IT resources, however that determination is not inferred by this work and is outside of the scope of this document. The anticiaption is that this work will adopt a stabdardised, genric representation for relationships when it becomes available.*

The manageability information model for this topic has a name **ManageableEndpointRelationships** which is defined in the **urn:wsmr:webservice:endpoint:relationships:manageability** namespace.

Following diagram shows the endpoint relationships topic model. The details are described in the following subsections.



1.3.5.2 Properties

- **relationships**
  This property is an array of known relationships for the source resource.

- **availableRelationships**
  This property is an array of available relationships types that can be provided by the resource.

- **availableRelationshipEvents**
  This property is an array of event identifiers that are available for relationships defined for the resource. Whenever the value of the target or source references are change, a relationshipTargetChange event is emitted.

1.3.5.3 Operations

- **QueryRelationships({<relationship query>})**
  Returns an array of relationships based on the supplied query parameters. For example it may look for relationship with a specified name, a specified relationship type, or a specified reference as a target or source.

1.3.5.4 Events

- **relationshipTargetChange**
  The event occurs whenever a relationship target changes.

1.3.6 Extensibility of Manageability

Web service providers may want to extend this information model and add custom manageability information unique to the particular Web service or its environment.

There are several ways to extend the information:

1) Add additional properties, operations and events into the interface extensions of the appropriate existing manageability topic interface through inheritance. Update the property classification lists appropriately.

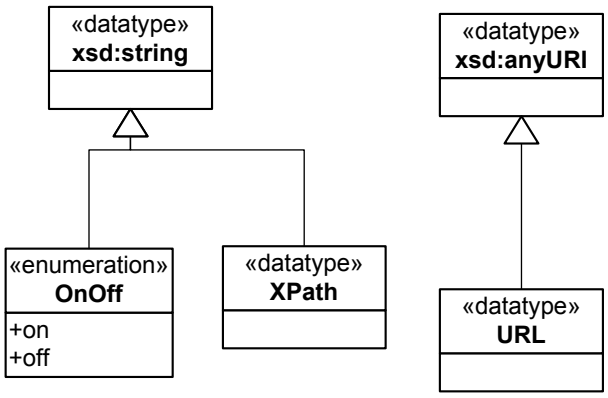
2) Create new topics if the new manageability extensions do not fit into an existing topic. For example, an additional topic may be “security” or “provisioning”. When a new topic is added, it should include the appropriate aspects (i.e. properties, operations, and events) applicable to the topic. Update the property classification lists appropriately.

3) Create additional properties, operations, and events directly into the interface of the manageability interface for the Web service. Update the property classification lists appropriately.

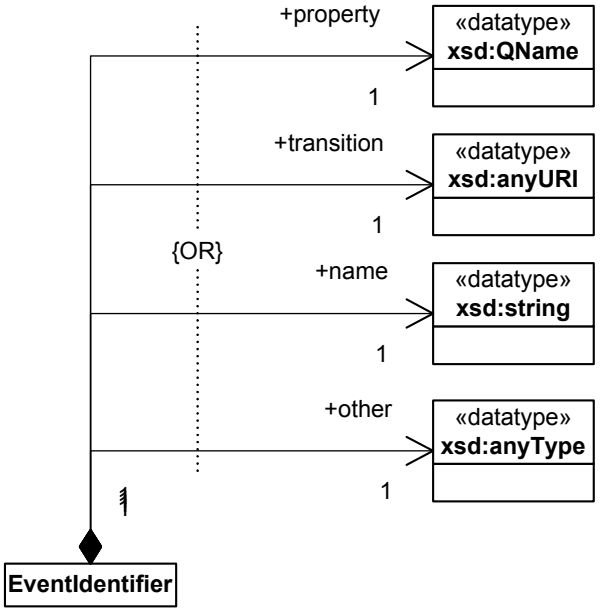
2 Web Service Endpoint Manageability Model Details

2.4 Common Data Types Used in the Model

OnOff, **XPath**, **URL** are defined as follows.



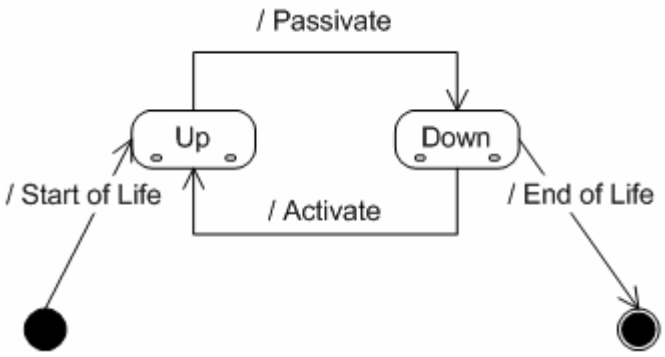
EventIdentifier is defined as follows.



2.5 Web service endpoint State Model

A Web service lifecycle is expressed in the state transition diagrams below. There are two separate transition paths: the ***endpoint lifecycle*** and the ***request processing***.

2.5.1 Lifecycle



States

- **UP** (compound) – a service is capable of accepting and processing requests (i.e. available).
- **DOWN** (compound) – a service is not capable of accepting any requests (i.e. not available).

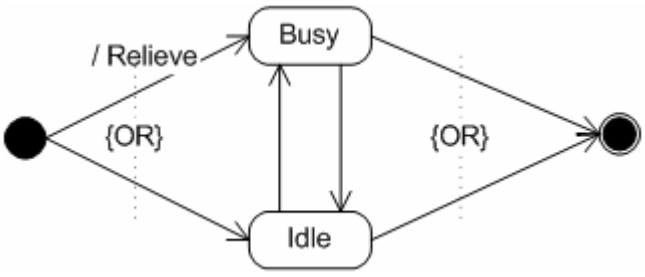
Transitions

- Start of Life (**SOL**) – a service starts its life in UP state.
- End of Life (**EOL**) – a service ends its life from DOWN state.
- **Activate** – a service can become available which transitions it from DOWN to UP state.
- **Passivate** – a service can become unavailable (for many reasons, see Down Substates) which transitions it from UP to DOWN state.

State Transition Table

Action	From State	To State	Sub-State	Comment
Activate	None - SOL	UP	IDLE	Instantiated and able to accept requests
Activate	None - SOL	DOWN	STOPPED	Instantiated but not able to accept requests
Passivate	UP	None - EOL	-o-	Destroyed
Passivate	DOWN	None - EOL	-o-	Destroyed

Up Substates



States

- **IDLE –** a service is not processing any requests currently.
- **BUSY –** a service is processing requests currently.

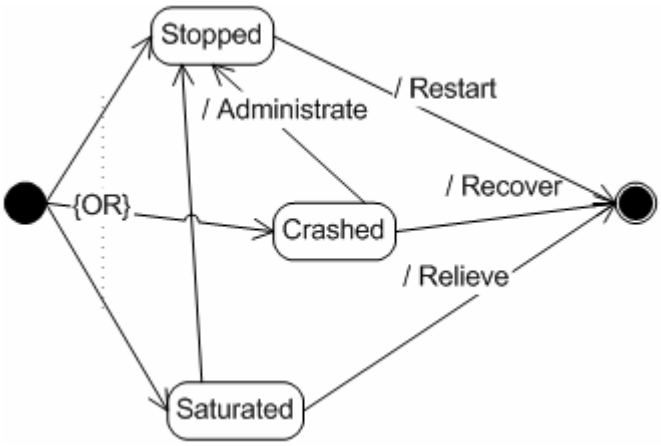
Transitions

- A service normally enters UP state in IDLE state.
- **Relieved** – a Service enters UP state in BUSY state if it was relieved (from DOWN/SATURATED, see Down Substates).
- A service may transfer from IDLE to BUSY and vice versa.
- A service exits UP state from either IDLE or BUSY.

State Transition Table

Action	Super State	Current State	End State	Comment
Accepts Request	UP	IDLE	BUSY	Currently PROCESSING requests
Completes all processing	UP	BUSY	IDLE	All requests completed (either FAILED or PROCESSED)

Down Substates



States

- **STOPPED –** a service was intentionally stopped (e.g. for administrative purposes).
- **SATURATED –** a service has exhausted its resources and cannot accept any new requests.
- **CRASHED –** a service is unavailable because of an internal malfunction (e.g. environmental problem).

Transitions

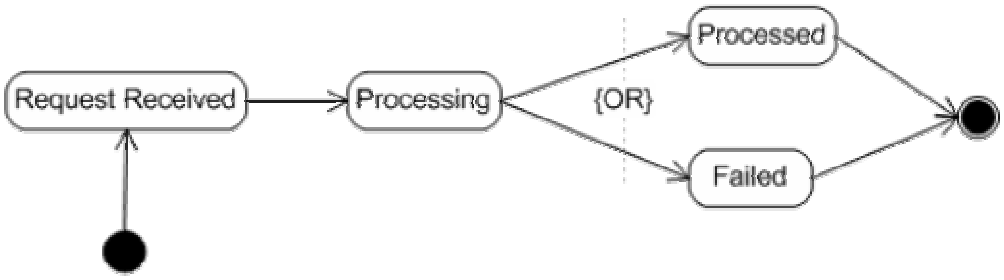
- A service enters DOWN state in either a STOPPED, CRASHED or SATURATED state.
- **Stopped -** a service may be stopped from any UP or DOWN state.
- **Relieve** – a service is relieved from saturation and exits a DOWN state (transitioning to an UP/BUSY state).
- **Recover** – a service was (automatically) recovered from malfunction and exits the DOWN state. It enters its UP state in the IDLE state
- **Restart** – a service was started again (manually) and exists a DOWN state. It naturally transitions to an UP/IDLE state.
- **Administrate** – a service can be stopped from being crashed (manually) for administration and maintenance.

1073 **State Transition Table**
1074

Action	Super State	Current State	End State	Comment
Manual Administration	UP	IDLE or BUSY	STOPPED	No longer accepting requests because of administrative action
Malfunction	UP	IDLE or BUSY	CRASHED	Malfunction in any UP state
Accepts Request	UP	BUSY	SATURATED	No longer accepting requests due to exhausted resources
Administrate	DOWN	CRASHED or SATURATED	STOPPED	No longer accepting requests because of administrative action
Processed or Failed Request(s)	DOWN	SATURATED	BUSY	Resources available after being exhausted such that requests can again be accepted
Recovery	DOWN	CRASHED	IDLE	Accepting requests again after a malfunction
Restart	DOWN	CRASHED or STOPPED	IDLE	Accepting requests again after being restarted
Manual Administration	DOWN	CRASHED	STOPPED	**NO LONGER CRASHED BUT NOT ACCEPTING REQUESTS**

1075

2.5.2 Request Processing



States

- **Request Received –** a service has accepted a request to perform one of its functional responsibilities.
- **Processing –** a service is doing some internal processing/execution to fulfill the requested function.
- **Processed –** a service successfully completed requested function returning results to the requestor.
- **Failed –** a service encountered an error and didn't complete the requested function, returning error to the requestor.

Transitions

- A service starts request processing when it accepts (receives) a request.
- A service starts execution (actual processing) after it received a request.
- A service transitions to either processed or failed state depending on the outcome of the processing/execution stage.
- A service exits request processing from either processed or failed state (which are mutually exclusive according to the previous transition).

State Transition Table

Action	Current State	End State	Comment
Accepts Request	-o-	REQUEST RECEIVED	Request received by the Service
Process the Request	REQUEST RECEIVED	PROCESSING	Commence execution of service function based on request received
Completion of processing	PROCESSING	PROCESSED	Successful completion of function executed based on the request received
Failure in processing	PROCESSING	FAILED	Unsuccessful completion of function executed based on the request received

2.6 Web service endpoint Relationship Model

There could be various mechanisms to discover and manage Web service relationships. In the previous sections, we have described how relationships are defined, but not how they are discovered or managed. This could be achieved in 3 ways;

- Associating at "runtime" by a manageability capability that returns relationships for a specific Web service, e.g. invoking an interface such as **getRelationships**.
- Associating at "design time" by declaring relationships in the WSDL documents of Web services, e.g. introspection of the WSDL.
- Associating by external "discovery" of Web service relationships that exist between Web services, e.g. asking a container for an external service registry or introspection of a UML model.

2.6.1 Manageability Capabilities

The Common Base Relationship model (see Appendix A) defines the manageability capabilities concerning relationships. The model is only slightly extended for the specific resource type Web service endpoint and is detailed in section 1.3.5.

***Note**: The model defined here of relationships is specific to the manageability of Web service endpoints and is an extension of the Common Base Relationship (CBR) model (see Appendix A). The CBR model may be suitable for describing relationships between other IT resources, however that determination is not inferred by this work and is outside of the scope of this document. The anticiaption is that this work will adopt a stabdardised, genric representation for relationships when it becomes available.*

2.6.1.1 Web service Endpoint Relationship Types

As detailed in section 1.3.5 the relationship types are identified using the following URIs.

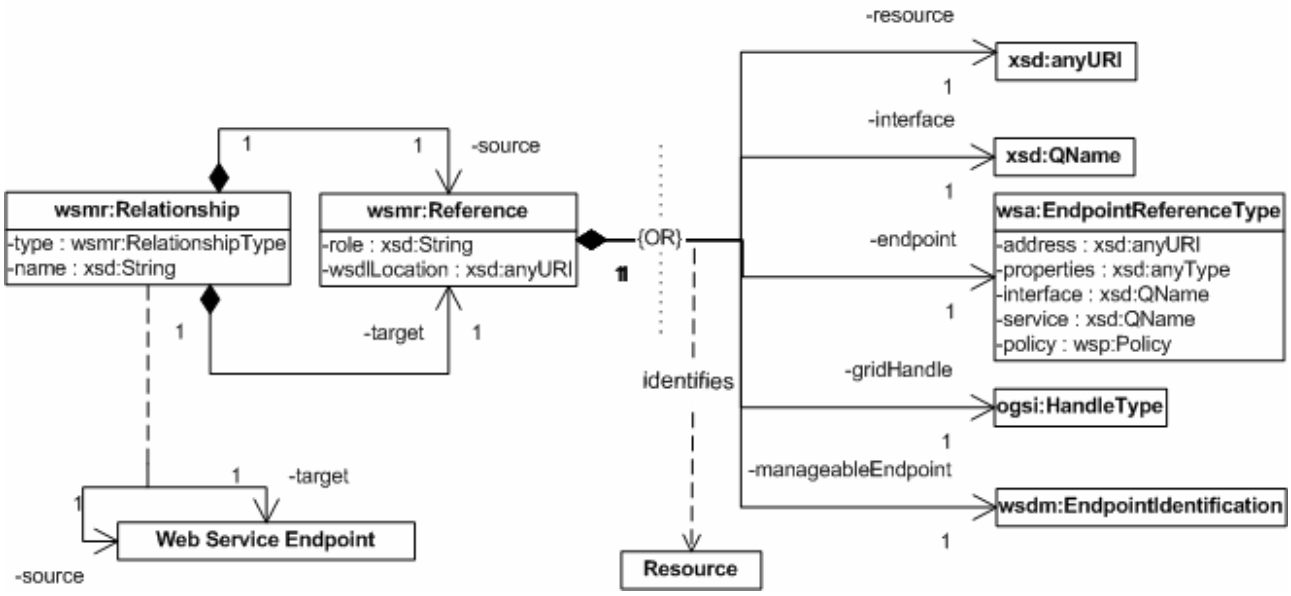
urn:wsmr:webservice:endpoint:dependency:uses
urn:wsmr:webservice:endpoint:change:compatibleWith

```
<xsd:simpleType name="RelationshipType">
  <xsd:restriction base="wsmr:AnyRelationshipType">
    <xsd:enumeration
     value="urn:wsmr:webservice:endpoint:dependency:uses"/>
     <xsd:enumeration
     value="urn:wsmr:webservice:endpoint:change:compatibleWith"/>
 </xsd:restriction>
</xsd:simpleType>
```

2.6.1.2 Web service Endpoint Relationship Description

Relationships are declared between two elements (of type wsmr:Reference) and include the type and name of the relationship.

The relationship model for Web service endpoints is shown below.



The model above is based on the CBR model defined in Appendix A. The only difference is the wser:EndpointReference type is an extension of wsmr:Reference (A 2.1) that allows identifying particular resources that a Web service endpoint may be related to.

wser:EndpointReference

This complex type defines references specific to a Web service endpoint. It is a specialization of the general reference to a resource. Note that any of the elements in the choice group can also be identified by a URI, and therefore, optional <reference> element of the wsmr:Reference type may also be included, but not required.

```
<xsd:complexType name="EndpointReference">
<xsd:extension base=”wsmr:Reference”>
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="interface" type="xsd:QName/>
         <xsd:element name="endpoint"
type="wsa:EndpointReferenceType"/>
         <xsd:element name="gridReference" type="ogsi:HandleType"/>
         <xsd:element name="resource" type="xsd:anyURI"/>
         <xsd:element name="managebleEndpoint"
type="wsdm:EndpointIdentification"/>
      </xsd:choice>
   </xsd:sequence>
</xsd:extension>
</xsd:complexType>
```

The following further describes the contents of this type.

```
<wser:reference>
      <wsmr:role>xsd:string</wsmr:role>?
      <wsmr:resource>xsd:anyURI</wsmr:resource>?
```

```
      (<wser:interface>xsd:QName</wser:interface> |
       <wser:endpoint>wsa:EndpointReferenceType</wser:endpoint> |
       <wser:gridHandle>ogsi:HandleType</wsmr:gridReference> |
      <wsmr:managebleEndpoint>wsdm:EdnpointIdentification
      </wsmr:resource>)
</wser:reference>
```

/wser:reference/wser:endpoint
This optional element (of type wsa:EndpointRefererenceType) defines the endpoint the reference refers to.

/wser:reference/wser:interface
This optional element (of type xsd:QName) defines the interface the reference refers to.

/wser:reference/wser:gridHandle
This optional element (of type ogsi:HandleType) defines the entity the reference refers to.

/wser:reference/wser:manageableEndpoint
This optional element (of type wsdm:EndpointIdentification) defines the manageable endpoint the reference refers to (see WS-Manageability specifications for more details on this type).

As defined above, a Web service endpoint references can be one of the following:

- An Endpoint Reference (wsa:EndpointReferenceType)
- An Interface (xsd:QName)
- A Grid Handle (ogsi:HandleType)
- A Resource (xsd;anyURI)

Endpoint References
As WSDL 1.2 becomes better defined, the endpoint referencing scheme it adopts is likely to form the basis of how this specification should define the structure of endpoint references. At this time, the WS-Addressing offers the clearest definition and can be used within this document to define the structure of an endpoint reference (http://xml.coverpages.org/ws-addressing20030313.pdf).

Interface References
Interface references are made by specifying the QName of the interface defined within the context of the WSDL document that describes a service.

Grid Service Handle
A Grid Service Handle (GSH) is an abstract naming construct defined within the Open Grid Services Infrastructure specification. They are used to name service instances but do not contain enough information to actually communicate with a grid service instance. To communicate with a grid service instance a GSH must be

resolved to a Grid Service Reference (GSR) that does contain the information required to communicate with a specific grid service instance ().

This mechanism allows for a level of indirections that makes distributed systems more flexible and supports the relationship model we defined, in terms of abstract relationships (e.g. interface or GSH).

Manageable Endpoint
See definition of the manageable Web service endpoint identification topic (1.3.1)

2.6.2 Manageability through WSDL Declarative Relationships

WSDL declarative relationships can be defined at the interface level, service level, and at the endpoint level and can be mixed. For example, an service defined in WSDL can also define its relationship to other interfaces or concrete endpoints.

Relationships when declared between two interfaces are similar to partner link type definitions in BPEL 1.1 However, each interface relationship is declared using the CBR schema defined in Appendix A (wsmr:Relationship in A 2.2, wsmr:Reference in A 2.1 and wser:EndpointReference in 2.6.1.2).

The following example shows how the relationship schema is used to define a dependency relationship between a travel agency service and services implementing airline reservation interfaces. The relationship is expressed in WSDL using WS-Policy as an example of a possible approach.

```
<definitions name="AnInterfaceDependencyExample"
targetNamespace="http://www.Travel.com/bookTrip/1.0/"
xmlns:tns="http://www.Travel.com/bookTrip/1.0/"
xmlns:fs="http://www.Airline.com/flightReservation/2.0/" … >
. . .
<wsp:Policy Name="BookTripPolicy">
<wsmr:relationships>
<wsmr:relationship>
<wsmr:type>urn:wsmr:webservice:endpoint:dependency:uses</wsmr:type>
<wsmr:name>Using a flight reservation service</wsmr:name>
<wsmr:target>
<wsmr:role>Flight Reservation Provider</wsmr:role>
<wser:interface>fs:FlightReservationPortType</wser:interface>
</wsmr:target>
</wsmr:relationship>
</wsmr:relationships>
</wsp:Policy>
. . .
<service name="BookTrip" wsp:PolicyRefs="tns:BookTripPolicy">
. . .
</service>
</definitions>
```

2.6.3 Manageability through Infrastructure Services

The CBR model for describing and managing relationships (Appendix A) also supports that the relationships in general become an independently defined element of the Web services architecture. This would allow for relationship management at an infrastructure service level.

Essentially it means that CBR could be used as a base model that is extended in terms of manageability interfaces for modification and discovery of relationships, etc. An infrastructure service (rather than a resource) may offer the extended relationship manageability interfaces.

3 Web Service Endpoint Manageability Model Representation

The models shown above can be rendered in WSDL 1.1, GWSDL, and WSDL 1.2 when it is available. It can also be used to create appropriate CIM models. The WSDL and GWSDL representations are in the accompanying Web Service Manageability – Representations document.

The representations document shows how the model (and topic sub-models) can be represented as WSDL 1.1 and GWSDL interfaces. Some of the techniques used may be useful in defining the manageability of resources other than Web services. Specifically, the development of common, base, and Web service endpoint specific interfaces for manageability topics. However, it is beyond the scope of this document to define how other manageability interfaces should be normatively rendered in WSDL. When the WSDM "Management Using Web Services" specification has been defined, the functions defined by the base portTypes will move to it and the remaining portTypes in this companion document will change, or new normative WSDL and GWSDLs will be defined, to align with the "Management Using Web Services" specification.

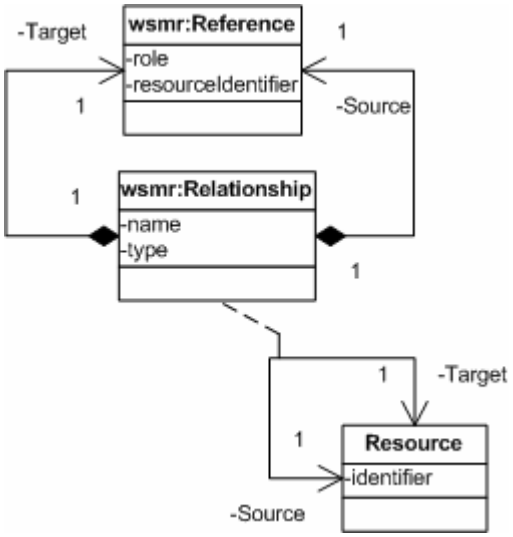
The GWSDL representations leverage existing OGSI and CMM functionality even where equivalent capability is not available in the WSDL 1.1 representations at this time.

Appendix A: Common Base Relationship Model

The Common Base Relationship Model (CBR) defines a general mechanism for describing types of relationships that can exist between IT resources. The general model is offered, such that it can be used to represent relationships between similar or disparate resources in the form of references. Each Relationship defines a link, modeled as an association, between exactly two Resources (a resource can, of course be related to many other resources, but each link is modeled as a separate relationship). The two resources in a relationship are modeled as References, one the *source* and the other the *target*. Each Reference identifies the resource as the source or target and the role it plays within the relationship.

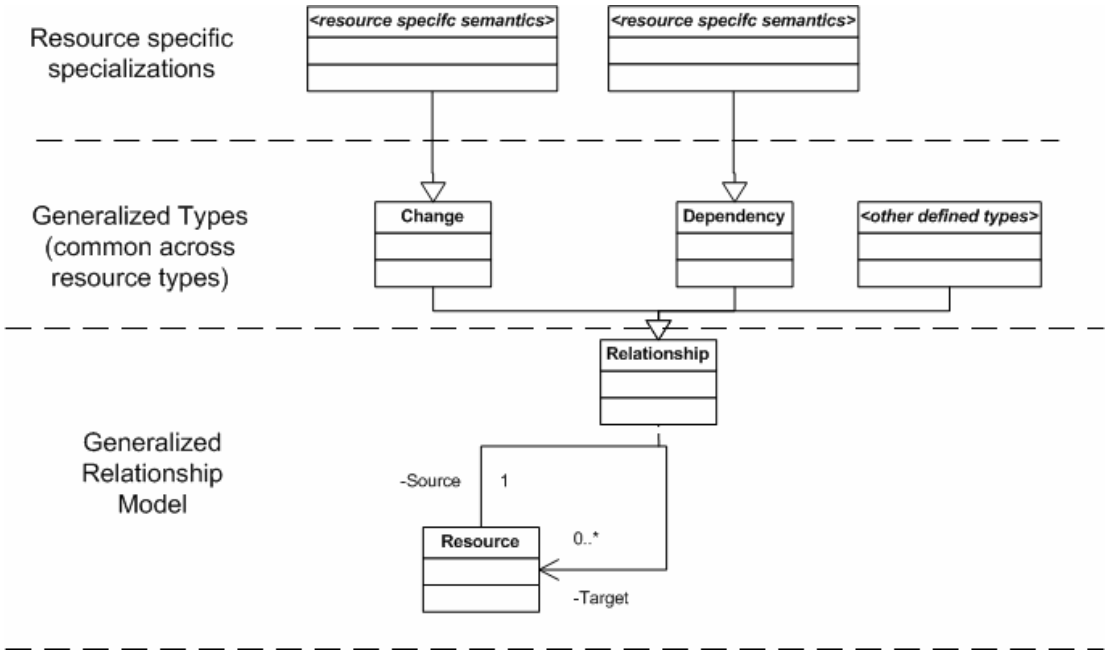
***Note**: Where a source Reference is not specified the Web service endpoint, **advertising** the relationship, is deemed to be the source. Where relationships need to be bidirectional new realtionship types can be defined (eg UsedBy) and the role element used to determine semantics for the traversal of the linkage (eg Flight booking provider).*

***Note**: The CBR model may be suitable for describing relationships between other IT resources, however that determination is not inferred by this work and is outside of the scope of this document. The anticiaption is that this work will adopt a stabdardised, genric representation for relationships when it becomes available.*



A.1 Common Relationship Types

The semantics of a relationship are defined by its type. The types of relationships identified based on manageability requirements and can be defined at two levels, generically where the semantics of the relationship are consistent across resources and between resource type and specifically where the semantics are definitive to the resource types being related.



Generalized types of relationships are shown in the diagram above as examples, that we expect to be defined by the Common Management Model (CMM) working group or the OASIS Web Services distributed Management (WSDM) technical committee as generic relationship types.

***Dependency**: A dependency category represents the need of one resource by another. For example, **Uses** is a type of dependency that states one resource directly exploits the services provided by the other resource. A software component **uses** another if it invokes an interface provided by that component or a software component **uses** a file if it reads or writes to it.*

***Change**: The change category represents relationships where a change has occurred. For example, **CompatibleWith** is a type of change relationship that states one resource maintains the same function a previous version, and can be used the same way.*

These relationship types are identified using URNs, using the following structure:

```
urn:wsmr:<resource type/name identifier>:<generic relationship
type>:<resource specific relationship type>
```

For example:

```
urn:wsmr:webservice:endpoint:change:compatibleWith
```

where **webservice** is the resource type, further qualified by **endpoint,** the generic relationship type is **change** and the specific relationship type is **compatibleWith**.

```
<xsd:simpleType name="AnyRelationshipType">
```

```
  <xsd:restriction base="xsd:AnyURI"/>
</xsd:simpleType>

<xsd:simpleType name="RelationshipType">
  <xsd:restriction base="wsmr:AnyRelationshipType">
    <xsd:enumeration value="……"/>
    <xsd:enumeration value="……"/>
   </xsd:restriction>
</xsd:simpleType>
```

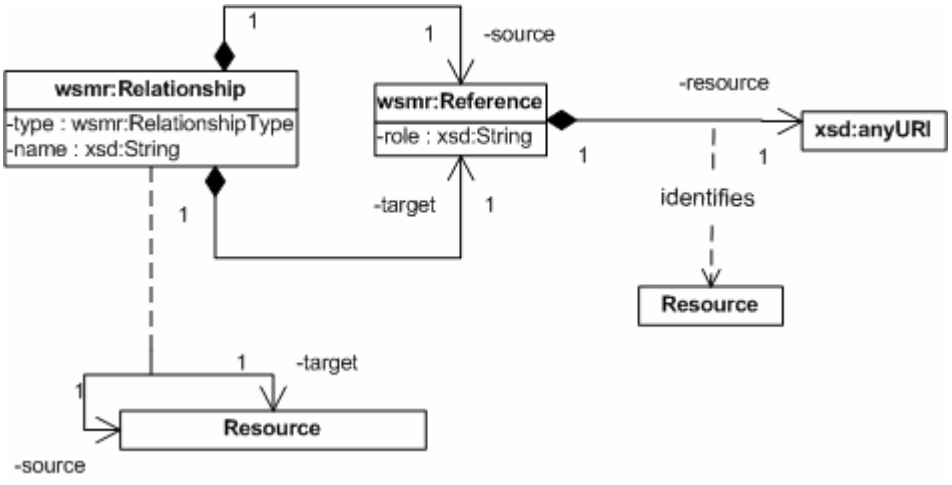
A.1.1 Constraints on Common Relationships

Certain constraints can be defined as part of the semantic definition of a relationship type. For example, generic or resource specific relationships could be defined as uni-directional where bi-directional a relationship could lead to non-resolvable circular relationships.

A.2 Common Relationship Description

Relationships are declared between two elements (of type wsmr:Reference) and declare the type and name of the relationship.

The Common Relationship Model for is shown below.



A.2.1 wsmr:Reference

The wsmr:Reference type defines a reference to a resource in a Relationship. The following schema definition describes this type.

So that resource, the reference refers to, can be specified in different ways, resourceGroup is defined as an abstract container element.

```
<xsd:complexType name="Reference">
```

```
   <xsd:sequence>
   <xsd:element name="role" type="xsd:string" minOccurs="0"/>
   <xsd:element name="resource" type=”xsd:anyURI” minOccurs=”0”/>
   <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
   </xsd:sequence>
</xsd:complexType>
```

This allows the resourceGroup element to be comprised of variable content. Variable content will be defined as substitutable elements, specific to the relationship type or resources being related. In the model seen above there is only one element defined for the variable content, but in later sections, we will see that the variable content could be a restricted choice of many different types.

/wsmr:reference
This represents a referenced entity in a relationship and specifies a resource and the role the reference plays in the relationship.

/wsmr:reference/wsmr:role
This required element (of type xsd:String) specifies the role this reference plays in a relationship.

/wsmr:reference/wsmr:resource
This is the URI that identifies the resource this reference points to.

A.2.2 wsmr:Relationship

The wsmr:Relationship type defines a Relationship type. The following schema definition describes this type.

```
<xsd:complexType name="Relationship">
   <xsd:sequence>
      <xsd:element name="type" type="wsmr:AnyRelationshipType"
minOccurs="0"/>
<xsd:element name="name" type="xsd:string" minOccurs="0"/>
      <xsd:element name="source" type="wsmr:Reference"
minOccurs="0"/>
      <xsd:element name="target" type="wsmr:Reference"/>
<xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
   </xsd:sequence>
</xsd:complexType>
```

The following further describes the contents of this type.

```
<wsmr:relationship>
      <wsmr:type>wsmr:AnyRelationshipType</wsmr:type> ?
      <wsmr:name>xsd:string</wsmr:name> ?
      <wsmr:source>wsmr:Reference</wsmr:source> ?
      <wsmr:target>wsmr:Reference</wsmr:target>
</wsmr:relationship>
```

/wsmr:relationship
This represents a type of relationship that can exist between two elements (of type reference), is named, and defines the type of relationship it represented between the two elements.

/wsmr:relationship/wsmr:type
This required element (of type xs:AnyURI) represents some element of type wsmr:RelationshipType. (see Relationship Types).

/wsmr:relationship/wsmr:name
This optional element (of type xsd:String) specifies the name of the relationship being specified.

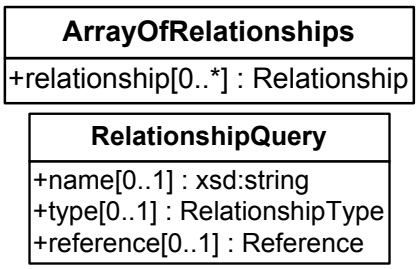
/wsmr:relationship/wsmr:source
This optional element (of type wsmr:Reference) specifies the [endpoint] property and the [role] property the source reference plays within the relationship.

/wsmr:relationship/wsmr:target
This required element (of type wsmr:Reference) specifies the [endpoint] property and the [role] property the target reference plays within the relationship.

A.3 Common Relationship Manageability Model

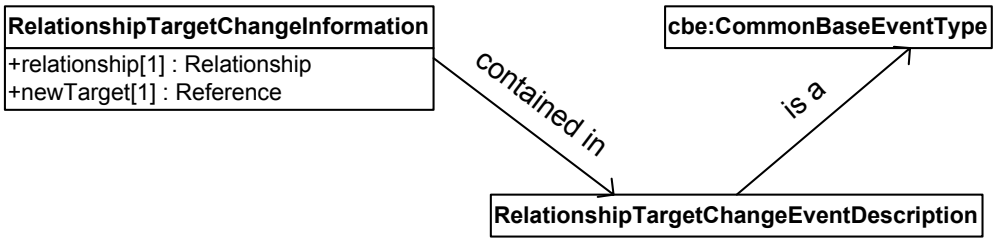
The relationship representation in UML and XML was given in section A.2. This section defines a few additional data types and an interface model for accessing the manageability of relationships capability.

Following additional data types are used in the manageability model. All additional data types specific to this sub model are defined in the **wsmr** namespace.



RelationshipQuery specifies a set of parameters that may be provided to query a set of resource relationships. The parameters define search criteria.

Following event description data types are defined for relationships.



The manageability information model for this topic has a name **ManageableRelationships** which is defined in the **urn:wsmr:common:relationships:manageability** namespace.

Following diagram shows the relationship interface model. The details are described in the following subsections.

ManageableRelationships
-relationships[1] : ArrayOfRelationships -availableRelationships[0..*] : RelationshipType -availableRelationshipEvents[0..*] : EventIdentifier «event» #relationshipTargetChange[1] : RelationshipTargetChangeEventDescription
+QueryRelationships(in query : RelationshipQuery) : ArrayOfRelationships

A.3.1 Properties

- **relationships**
  This property is an array of known relationships for the source resource.

- **availableRelationships**
  This property is an array of available relationships types that can be provided by the resource.

- **availableRelationshipEvents**
  This property is an array of event identifiers that are available for relationships defined for the resource.

A.3.2 Operations

- **QueryRelationships({<relationship query>})**
  Returns an array of relationships based on the supplied query parameters. For example it may look for relationship with a specified name, a specified relationship type, or a specified reference as a target or source.

A.3.3 Events

- **relationshipTargetChange**

1519 The event occurs whenever a relationship target changes.
1520

Appendix B: Web service Relationship Examples

B.1 Web service endpoint Dependency:Uses Representation

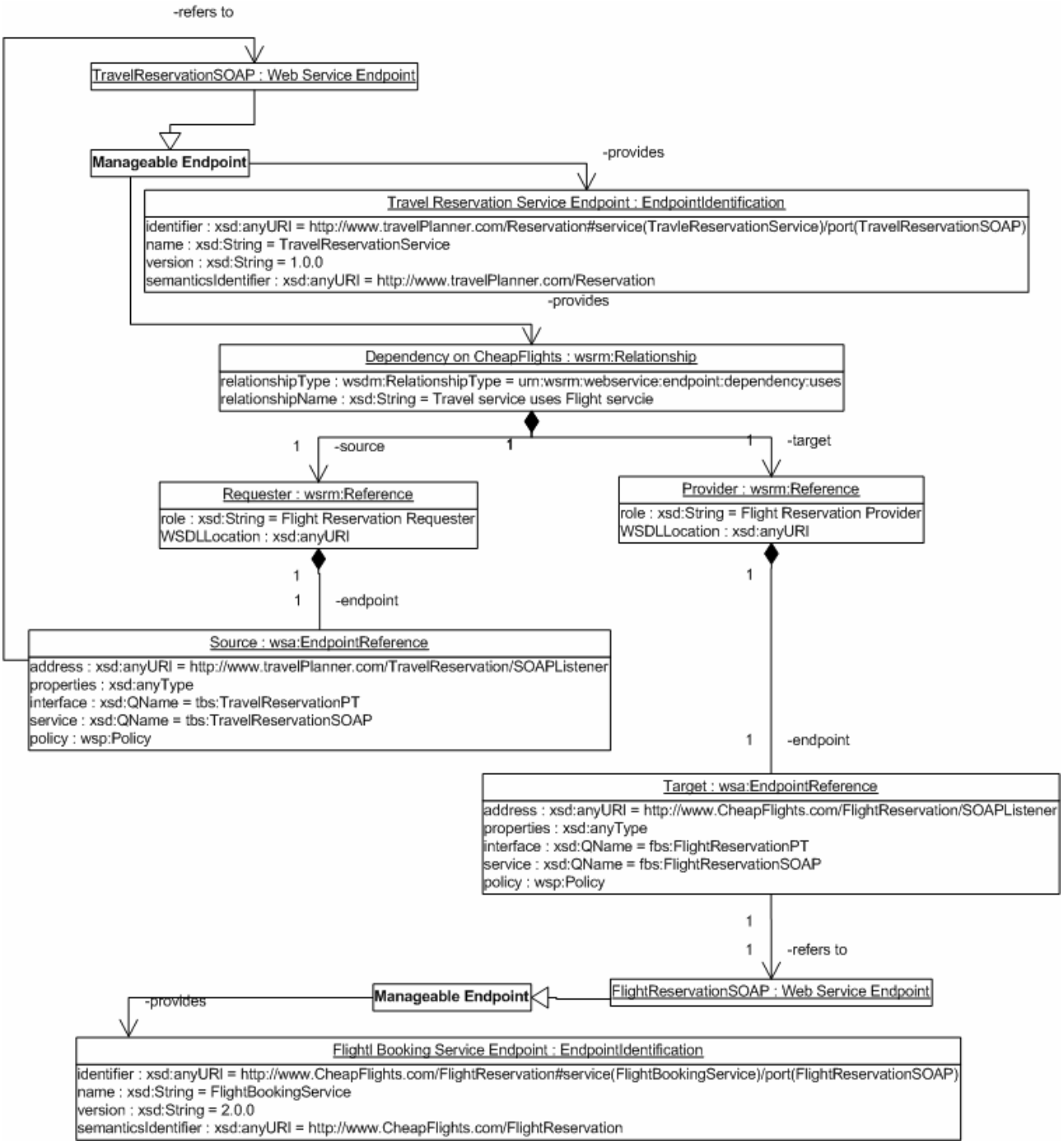
Based on the general model for expressing relationships a dependency can be defined in two ways, as per the model, at the interface level, and at the endpoint level. Declaring relationships at the interface level for dependencies, allows for visibility into relationships even where Web service endpoint to be utilized are resolved dynamically at runtime. The resolution would likely be in the form of a Web service endpoint reference that contains the relevant information required to invoke the Web service endpoint (binding to be used and the endpoint address). The Web service endpoint reference is also, the way in which endpoint level relationships for a web service endpoint are defined.

B.1.1 Endpoint defined Dependency:Uses

A dependency that is defined in terms of an endpoint reference implementing an interface (early binding) is called an endpoint dependency. The structure of the endpoint reference could be rendered using WS-Addressing (http://xml.coverpages.org/ws-addressing20030313.pdf), Arthur Ryman's R085 proposal (http://lists.w3.org/Archives/Public/www-ws-desc/2002Dec/att-0021/01-URI-References.html), the alternate Oracle submission to the WSDL working group at the W3C (http://lists.w3.org/Archives/Public/www-ws-desc/2003Jun/att-0005/WS-Ref3.pdf). The specification for these concepts as they relate to Web Services is still evolving, and we expect normative definitions for them to emerge in the future.

Example

The following example shows how the relationship model described in section 2.6.1.2 can be used to define a uses relationship between the TravelBooking service of a TravelReservation service, and an endpoint reference to a Flight service. The model is shown below followed by an XML rendition based on the schema defined.


-refers to
TravelReservationSOAP : Web Service Endpoint
Manageable Endpoint
-provides
Travel Reservation Service Endpoint : EndpointIdentification
identifier : xsd:anyURI = http://www.travelPlanner.com/Reservation#service(TravleReservationService)/port(TravelReservationSOAP)
name : xsd:String = TravelReservationService
version : xsd:String = 1.0.0
semanticsIdentifier : xsd:anyURI = http://www.travelPlanner.com/Reservation
-provides
Dependency on CheapFlights : wsrm:Relationship
relationshipType : wsdm:RelationshipType = urn:wsrm:webservice:endpoint:dependency:uses
relationshipName : xsd:String = Travel service uses Flight servcie
1
-source
1
1
-target
Requester : wsrm:Reference
role : xsd:String = Flight Reservation Requester
WSDLLocation : xsd:anyURI
Provider : wsrm:Reference
role : xsd:String = Flight Reservation Provider
WSDLLocation : xsd:anyURI
1
1
-endpoint
1
Source : wsa:EndpointReference
address : xsd:anyURI = http://www.travelPlanner.com/TravelReservation/SOAPListener
properties : xsd:anyType
interface : xsd:QName = tbs:TravelReservationPT
service : xsd:QName = tbs:TravelReservationSOAP
policy : wsp:Policy
1
-endpoint
Target : wsa:EndpointReference
address : xsd:anyURI = http://www.CheapFlights.com/FlightReservation/SOAPListener
properties : xsd:anyType
interface : xsd:QName = fbs:FlightReservationPT
service : xsd:QName = fbs:FlightReservationSOAP
policy : wsp:Policy
1
1
-refers to
-provides
Manageable Endpoint
FlightReservationSOAP : Web Service Endpoint
FlightI Booking Service Endpoint : EndpointIdentification
identifier : xsd:anyURI = http://www.CheapFlights.com/FlightReservation#service(FlightBookingService)/port(FlightReservationSOAP)
name : xsd:String = FlightBookingService
version : xsd:String = 2.0.0
semanticsIdentifier : xsd:anyURI = http://www.CheapFlights.com/FlightReservation

1549

The following is a schema rendered version of the UML example above.

```
<wsmr:relationship>
       <wsmr:type>urn:wsmr:webservice:endpoint:dependency:uses</wsmr:type>
       <wsmr:name>Travel service uses Flight service</wsmr:name>
       <wsmr:source>
             <wsmr:role>Flight Reservation Requester</wsmr:role>
             <wsmr:endpoint
xmlns:tbs="http://www.travelPlanner.com/TravelReservation">
                  <wsa:Address>
          http://www.travelPlanner.com/TravelReservation/SOAPListener
                  </wsa:Address>
                 <wsa:PortType>tbs:TravelReservationPT</wsa:PortType>
                  <wsa:ServiceName PortName="TravelReservationSOAP">
                     tbs:TravelReservation
                   </wsa:ServiceName>
            </wsmr:endpoint>
           <wsmr:wsdlLocation>
           http://www.travelPlanner.com/wsdl/TBS.wsdl
           </wsmr:wsdlLocation>
       </wsmr:source>
       <wsmr:target>
              <wsmr:role>Flight Reservation Provider</wsmr:role>
              <wsmr:endpoint
xmlns:fbs="http://www.CheapFlights.com/FlightReservation">
                   <wsa:Address>
          http://www.CheapFlights.com/FlightReservation/SOAPListener
                   </wsa:Address>
                  <wsa:PortType>fbs:FlightReservationPT</wsa:PortType>
                   <wsa:ServiceName PortName="FlightReservationSOAP">
                     fbs:FlightReservation</wsa:ServiceName>
              </wsmr:endpoint>
             <wsmr:wsdlLocation>
             http://www.CheapFlights.com/wsdl/FR.wsdl
             </wsmr:wsdlLocation>
      </wsmr:target>
</wsmr:relationship>
```

B.1.2 Interface defined Dependency:Uses

Interface dependencies are resolved to endpoints at runtime through an endpoint resolution mechanism (late binding). This type of declaration defines "I need a web service endpoint that will allow me to send and receive messages as defined in the interface I am identifying". The assumptions here are that the endpoint reference will give me enough information to send the message I create to the URI (endpoint defined in the endpoint reference).

Example

The following shows the same example used above (B.1.1) but describes the relationship between the TravelBooking service of a TravelReservation service, at an interface level on both sides of the relationship. The example is shown below as an XML rendition based on the schema defined.

```
<wsmr:relationship>
       <wsmr:type>urn:wsmr:webservice:endpoint:dependency:uses</wsmr:type>
       <wsmr:name>Travel service uses Flight service</wsmr:name>
```

```
       <wsmr:source>
         […  as above  …]
       </wsmr:source>
       <wsmr:target>
              <wsmr:role>Flight Reservation Interface</wsmr:role>
              <wsmr:interface>fbs:FlightReservationPT
             </wsmr:interface >
              <wsmr:wsdlLocation>
             http://www.CheapFlights.com/wsdl/FR.wsdl
             </wsmr:wsdlLocation>
      </wsmr:target>
</wsmr:relationship>
```

B.2 Web service endpoint Change:compatibleWith Representation

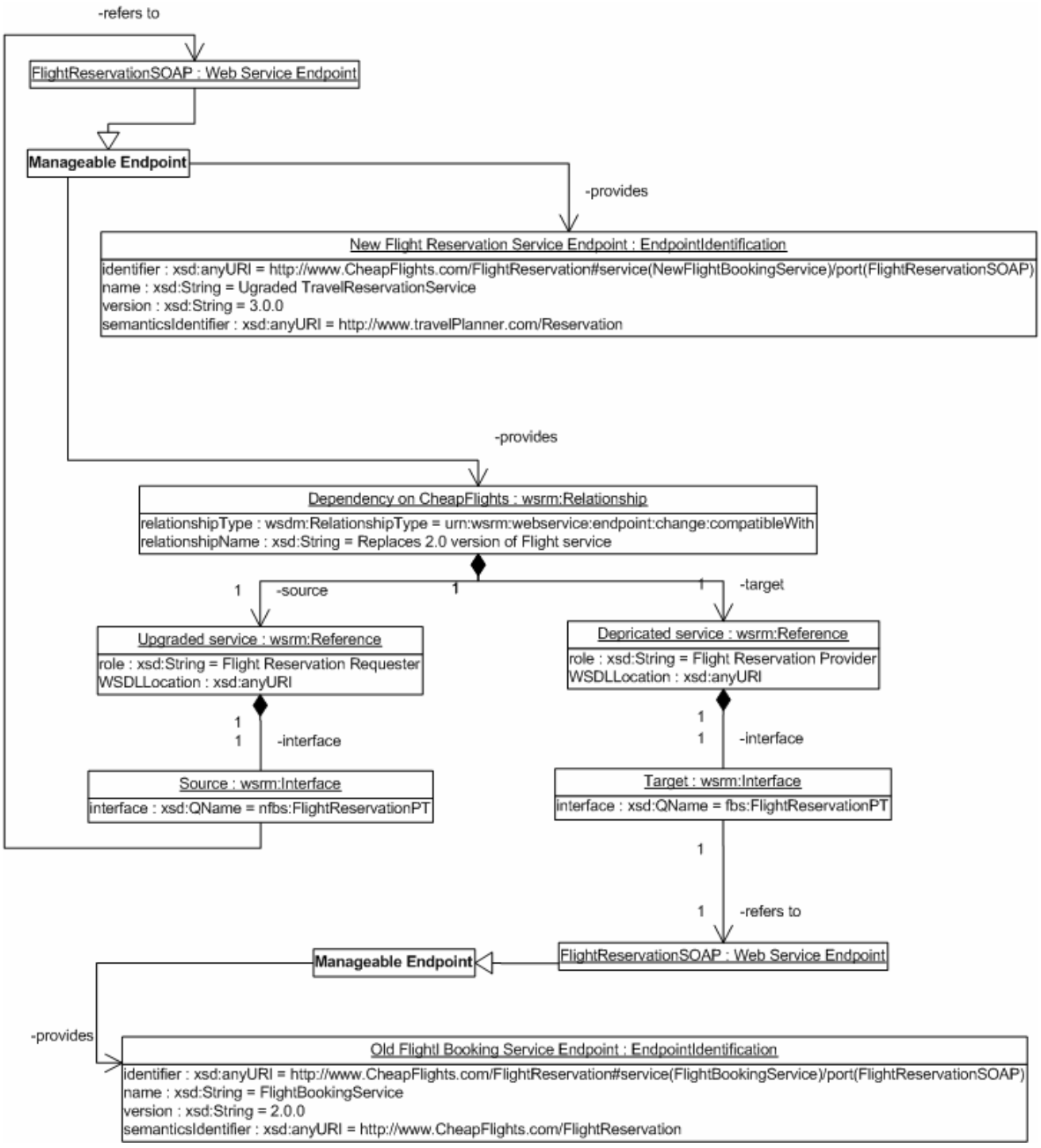
Based on the general model for expressing relationships, compatibility can be defined in two ways, at the interface level, and at the endpoint level**.**

B.2.1 Interface defined Change:compatibleWith

Declaring relationships at the interface level is more likely when declaring compatibility. Compatibility is defined such that messages constructed from the service description of one service can be sent to the related service without any required changes to the message construct, and most importantly in the service the requester receives.

Example

The following example shows how the relationship model described in section 2.6.1.2 can be used to define a compatibility relationship between a **new** Flight reservation service and the **original** Flight service. The model is shown below followed by an XML rendition based on the schema defined.



The following is a schema rendered version of the UML example above.

```
<wsmr:relationship>
 <wsmr:type>urn:wsmr:webservice:endpoint:chnage:compatibleWith
 </wsmr:type>
 <wsmr:name>Travel service upgrade</wsmr:name>
       <wsmr:source>
             <wsmr:role>Upgraded Flight Reservation Interface</wsmr:role>
             <wsmr:interface>nfbs:FlightReservationPT</wsmr:interface >
             <wsmr:wsdlLocation>
              http://www.CheapFlights.com/wsdl/NFR.wsdl
              </wsmr:wsdlLocation>
       </wsmr:source>
```

```
       <wsmr:target>
             <wsmr:role>Depricated Flight Reservation Interface</wsmr:role>
             <wsmr:interface>fbs:FlightReservationPT</wsmr:interface >
             <wsmr:wsdlLocation>
              http://www.CheapFlights.com/wsdl/FR.wsdl
              </wsmr:wsdlLocation>
      </wsmr:target>
</wsmr:relationship>
```

Appendix C: Issues to this document

Issue #1 (by Igor Sedukhin):

Optionally there may be events from the start of a transition, in which case duration will be missing from the event information.

Current specification prohibits that.

Issue #2 (by Don Ferguson)

A set of specifications introduced in this specification should be replaced by appropriate standards for the Web services community at large. Including:

Event Polling (replacing CommonEventAccess portType)

Event Emission Control (replacing CommonEventControl portType)

Relationships Description (replacing relationships)

Relationships View (replacing manageableRelationship)

Runtime support of Attributes to represent property aspects (replacing CommonDataAccess portType)

Additional specifications are required to provide support for properties in WSDL.

Description support of Attributes to represent property aspects

Attribute metadata and annotation support for description and runtime

- property classification (replacing propertyList)
- volatility
- mutability