Web Services Distributed Management: Management of Web Services (WSDM-MOWS 0.5)

Committee Draft, 2 April 2004

Document identifier:
cd-wsdm-mows-0.5-20040402

Location:
http://docs.oasis-open.org/wsdm/2004/04/mows-0.5

Editors:
John DeCarlo, Mitre Corporation <jdecarlo@mitre.org>
Igor Sedukhin, Computer Associates <igor.sedukhin@ca.com>

Abstract:
Web Services Distributed Management (WSDM) specification, as declared in the committee charter **[Charter]**, defines management of any IT resource via Web services protocols (Management Using Web Services, or MUWS) and management of the Web services resources via the former (Management Of Web Services, or MOWS). This document is the part of WSDM specification defining MOWS.

Status:
This is a draft document and there is no guarantee any part of its content will appear in the final release specification. This document is updated periodically on no particular schedule. Send editorial comments to the editor.

Committee members should send comments on this specification to the wsdm@lists.oasis-open.org list. Others should subscribe to and send comments to the wsdm-comment@lists.oasis-open.org list. To subscribe, send an email message to wsdm-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 WSDM TC web page (http://www.oasis-open.org/committees/wsdm/).

Since this specification is not yet final, there are no errata available.

Table of Contents

1 Introduction

Web services are an integral part of the IT landscape, and, as such, are vital resources to many organizations. Web services may interact with other Web services and are used in business processes. Interacting Web services form a logical network which may span enterprise boundaries. Managing such a logical network is critical for organizations that use Web services to automate and integrate various internal functions, and deal with partners and clients electronically. To manage the Web services network, one needs to manage the components that form the network – the Web services endpoints. This part of WSDM specification addresses management of the Web services endpoints using Web services protocols **[MOWS-Reqs]**.

The *Management Of Web Services* (MOWS) specification is based on the concepts and definitions expressed in the *Management Using Web Services* specification (MUWS) **[MUWS]**. It is recommended that the reader is aware of the MUWS specification contents.

Definitions and examples in this document are based on the following specifications. It is recommended that the reader is aware of their contents.

- WS Architecture **[WS-Arch]**
- XML **[XML]**
- XML Namespaces **[XNS]**
- XML Schema **[XMLS]**
- SOAP **[SOAP]**
- WSDL **[WSDL]**
- WS-Addressing **[WSA]**
- WS-ResourceProperties **[WSRP]**

Section 3 and appendices D and E are *normative* specifications. The rest of the document is *non-normative*, and is provided as background and explanatory material.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT","SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in **[RFC2119]**.

1.2 Notational conventions

This specification uses an informal syntax to describe the XML grammar of the messages, property instances and event information making up the management interfaces. This syntax uses the following rules:

- The syntax appears as an XML instance, but the values indicate the data types instead of values.
- {any} is a placeholder for elements from some other namespace (like ##other in XML Schema).
- Characters are appended to attributes, elements, and {any} to indicate the number of times they may occur as follows: ? (0 or 1), * (0 or more), + (1 or more). No character

indicates exactly 1 occurrence. The characters [and] are used to indicate that contained items are to be treated as a group with respect to the ?, *, and + characters.

- Attributes, elements, and values separated by | and grouped with (and) are meant to be syntactic alternatives.
- ... is used in XML start elements to indicate that attributes from some other namespace are allowed.
- The XML namespace prefixes (defined below) are used to indicate the namespace of the element being defined

A full WSDL description of all interfaces and XML Schemas of all information elements are available in the appendices.

2 Overview of the Web service endpoint manageability

Management of Web services (MOWS) is a particular case of Management using Web services (MUWS) in which a resource is an element of the Web Services Architecture **[WS-Arch]**. This draft only addresses manageability of Web service endpoints.

The Web services concepts, according to the WSDL specification, are defined as follows. A service is an aggregate of endpoints each offering the service at an address and accessible according to a binding. A service has a number of interfaces that are realized by all of its endpoints. Each interface describes a set of named messages that could be exchanged and their format. Properly formatted messages could be sent to an endpoint's address in a way prescribed by the binding. A description (document, artifact) is composed of definitions of interfaces and services. A description may contain both or either of the definitions.

An IT resource may bear some functional (e.g. business) responsibilities such as, for example, placement of an order. That would constitute a functional capability with the distinct semantics of placing an order. A functional resource is a composition of such capabilities. An endpoint may provide access to the functional resource and in that case would offer those capabilities. Such an endpoint is called a functional endpoint. To offer a capability, an endpoint has to realize interfaces. An interface that represents a functional capability is called a functional interface. One capability may be represented by many interfaces (e.g. various ways of representing the same semantics for different groups of target users).

The MUWS manageability concepts are defined very similarly to the functional concepts (see the MUWS specification). According to MUWS, a manageable resource is a resource that is composed of a number of manageability capabilities, each represented by one or more manageability interfaces.

Management of Web services starts at an endpoint resource which, therefore, becomes a manageable resource, specifically called a manageable endpoint. The reason the endpoint is the basic element is that (1) anything behind an endpoint is a concrete implementation (e.g. an application hosted in a container), and (2) anything that builds on endpoints is a logical construct understanding of which has to be inferred from the realization of the endpoints that aggregate into it. This specification focuses on defining manageability of the Web service endpoints and the rest is out of scope of this document.

Because a manageable endpoint is a manageable resource, it composes a number of manageability capabilities. Some of the capabilities may be generic, as defined in MUWS, and some may bear semantics specific to MOWS. For example, metrics available on Web services endpoint resources only may be captured in a UML model named **EndpointMetrics** which can be represented (rendered) as an **EndpointMetrics** WSDL interface (portType) defined in the **http://docs.oasis-open.org/wsdm/2004/04/mows-0.5/wsdl** namespace. The UML model is an instance of the manageability capability concept and the WSDL interface description is an instance of the manageability interface concept. There could be other possible renditions of the same UML model in other interface representations.

The following UML diagram captures the MOWS concepts and their relationships as expressed above.

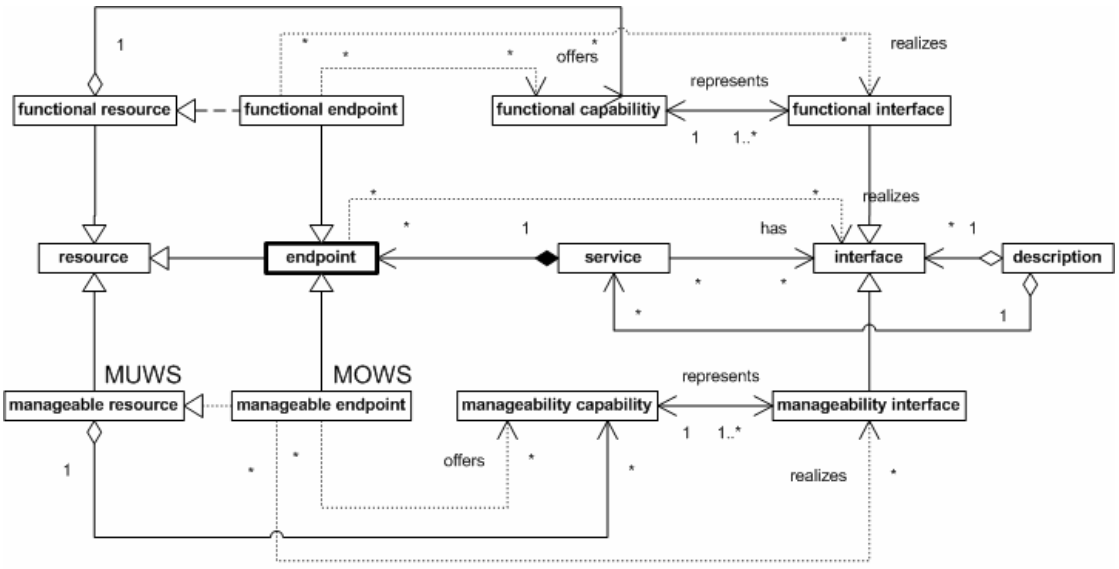

Figure 1. MOWS concepts and their relationships

2.1 Locus of implementation

MUWS concepts define that manageability of a given resource is accessible via one or more manageability endpoints which are Web service endpoints.

In the case that a resource IS an endpoint and, therefore, the manageable resource IS a manageable endpoint, the manageability endpoint MIGHT be the same as the manageable endpoint OR it might be different.

The following UML diagram formally captures the above statement.

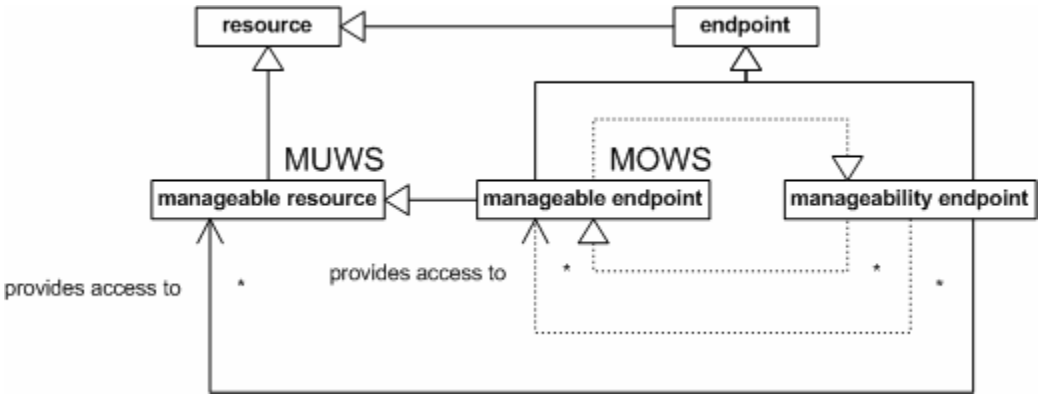

Figure 2. MOWS locus of implementation

2.2 Relationship to Management Using Web Services

The MUWS specification defines common manageability capabilities applicable to any resource, for example, a capability to expose any metrics is a common capability. MOWS specification defines manageability capabilities of a Web service endpoint, for example, a capability to expose specific metrics applicable to the endpoint. Both the common manageability capabilities and specific manageability capabilities can be equally composed into a manageable endpoint resource.

An endpoint manageability capability may depend on a common manageability capability. This dependency is optional, however. The dependency could be an explicit extension of a common capability to make it more specific. For example, the common manageable state capability currently represents an ability to express Available/Unavailable/Degraded states, and an endpoint manageable state capability may add an ability to represent IDLE/BUSY/STOPPED, and other endpoint-specific states. This could be expressed by an endpoint manageable state UML model (class) that extends the common manageable state UML model (class). There could be cases where extension is implicit. For example, a UML model of the endpoint manageable metrics capability could use some of the data types expressed in the common manageable metrics UML model (e.g. Counter data type), but the capability model itself does not have to mandate the extension of the whole common capability model. That is, the endpoint manageable metrics capability can be supported on its own without the need to support the common capability. There also could be cases when an endpoint manageability capability is a new one, available for Web service endpoints only, and there is no dependency on a common capability.

The following UML diagram formally captures the above statement.

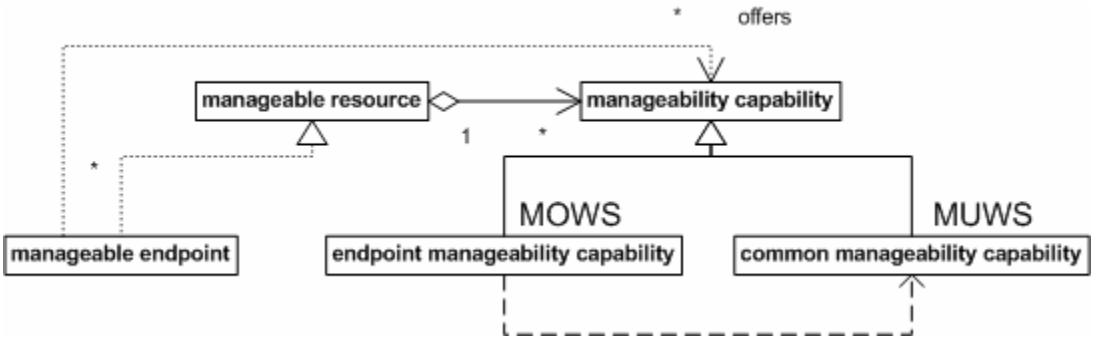

Figure 3. Relationship of MOWS and MUWS

2.3 Composability

A resource (such as, a disk) could be exposed as a Web service. For example: its read/write/seek function could be exposed as a service. WSDM specifications allows the resource and its service to be manageable in a standard and interoperable manner by defining manageability capabilities and interfaces of a resource and a service (a kind of a resource too).

Manageability capabilities and interfaces could be composed into the service that offers functions of the resource. For example, a Web service for a manageable disk resource would implement its functional interfaces and also could implement interfaces that allow disk management and management of the service that offers the disk functions.

Managers could easily discover such composition by inspecting the service description. Managers could take advantage of the composition of manageability by, for example, querying free disk space using disk manageability capability and, along with that, reading sectors from the disk using the functional service.

Composability makes it easy for managers to deal with resources exposed as Web services and also makes it easy for implementers of the resource services to offer a proper set of manageability capabilities.

The following diagram illustrates the composability feature described in the preceding paragraph.

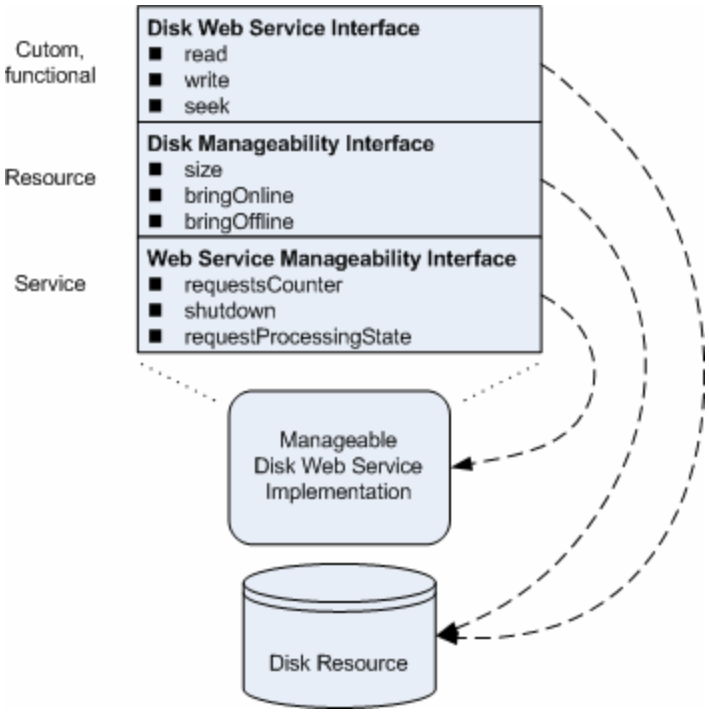

Figure 4. Composability

2.4 Responsibilities of the provider of manageability

The system providing manageability capabilities for a service must be aware of the configuration of the service from the caller's point of view. This *configuration* may be dependent upon external hardware or software options. Manageability may need to be implemented differently depending upon the requests made with respect to the caller's point of view.

Consider two examples. The first case is that of a hardware routed service. By this, we refer to the case where some hardware device offers up a service at, for example, http://external.example.com/theService. Upon receipt of messages for that URL, the device forwards the messages to any service from the set:

- http://s1.example.com/theService
- http://s1.example.com/theOtherService
- http://s2.example.com/yetAnotherService

These services are identical, providing access to the same underlying business resource.

If, say, a query regarding metrics were made regarding the service http://external.example.com/theService, it is the responsibility of the provider of manageability to aggregate the results from the three underlying services to provide a meaningful response.

A second example is one wherein a single service is known by two distinct names. In this case, consider the service at http://services.example.com/creditCheck. External to the Example Company, this service is known as "http://ourservices.example.com/creditCheck", while internally, this service is known as "http://extservices.example.com/creditCheck". However, in both cases, the underlying service is performed by the same machine, service, etc. The service itself is *aware* of the means by which it is addressed, and it adjusts itself appropriately.

In this case, the provider of manageability must be similarly aware of how a service was addressed. Queries regarding the two URL's must be accounted for separately, even though the underlying service is identical, quite possibly with the distinction between the two maintained only using different name servers.

2.5 Manageability at the Web service level

A Web service endpoint is defined as the implementation of a WSDL 1.1 portType with a given WSDL 1.1 binding at a given URL. In a WSDL1.1 document, it corresponds to a port element. There is no guarantee that only one endpoint corresponds to a given URL. This specification defines an endpoint as what is described by a <port> element in a WSDL 1.1 document.

WSDL 1.1 defines a service element as a collection of port elements. There is no requirement that these ports have anything in common in terms of portTypes, bindings or endpoint URLs. (Note that the current draft of the WSDL 2.0 specification requires that all ports in a service implement the same interface - the new name for portType.) Therefore, WSDL 1.1 defines a Web service as any collection of endpoints that one chooses to group together in a service WSDL 1.1 element. The same set of endpoints can be grouped at the same time in many permutations of services by WSDL authors. For visibility and other concerns, many WSDL documents may include descriptions of the same service with different endpoints. In certain cases, a WSDL document may include a description of a service with endpoints offered by different providers. In addition, other specifications can claim to define Web services, such as UDDI, that do not use the same mechanism.

Implementing management at the Web service level therefore offers challenges in terms of identifying services. It also offers implementation challenges, for example if all the endpoints in a service are not implemented in the same environment (e.g., one endpoint inside the firewall and one endpoint outside of the firewall). Also, in many cases managers want to manage Web services at the granularity level of the endpoint. For example, they need to know when one endpoint goes down and how many messages a specific endpoint has processed. At the same time, there are many cases where the manager wants to think at the Web service level and doesn't care about the endpoint. For example, a business manager using a business dashboard doesn't care whether the purchase orders arrive via the HTTP or the SMTP binding of the purchase service, or whether they arrive via the US server or its European mirror.

In recognition of these requirements, the WSDM MOWS specification defines manageability of endpoints as the base building block for managing Web services. It also ensures that information is available for the manager to reconstruct the service-level view that some users require. This

includes allowing a request by a manager of the list of WSDL documents which are known to the endpoint (to identify services in which this endpoint participates). It also includes allowing endpoints to establish relationships linking them as part of the same service. One way a manager can be allowed to access a set of endpoints (representing a service) as one entity would be through a collection mechanism. Finally, the MOWS specification will identify in a non-normative way, the capabilities of a service and how they can be derived from the capabilities of the endpoints that compose them.

2.6 Versioning concepts applied to Web services

It is expected that the interfaces and implementations of Web services, like all other information systems, will change over their lifetime. These changes need to be managed. Fortunately, Web services can draw upon several decades of refinement in the management of interfaces and the software that implements them. In particular, the following capabilities are needed:

- The ability to distinguish versions of Web services as they evolve over time, via some sort of version identification that can be used by a service provider and consumer.
- For the provider, the ability to identify the pieces and parts that comprise a single version. The pieces may be interface definitions, implementation components, security and management policies, etc. Each of these components may be separately versioned. A set of components that are consistent and work properly together constitute a “baseline” of the Web service that can be assigned an externally visible version identification.
- A means to proactively manage the change process. This involves:
    - The ability to describe the changes in individual components and aggregate those change descriptions to the Web service as a whole.
    - The ability to notify consumers of a Web service and communicate the schedule, nature, impact and details of changes.

The elements of the Web services architecture, expressed in WSDL, could be versioned. For example, the description, interface, service and endpoint could be defined in their own target namespaces that are not necessarily the same. The namespace differences represent that versions of those components may be different.

In this case, the difference is one of version. Therefore, Web services elements can optionally have version information that includes a version date and a version number in the form of a dotted notation: major/minor/release/build (e.g., 1.4.3.1230).

Each version optionally has one or more change descriptions that help enumerate the changes made since the last update. Each change description may be viewed as a document or a separate statement of some sort (e.g., "new interface was implemented"). These change descriptions are held separately for each Web service element because the elements can be changed independently of the others. This is the same idea as providing a description when a new version of a file is checked into a version control system.

The following UML diagram formally captures the above description of versioning.

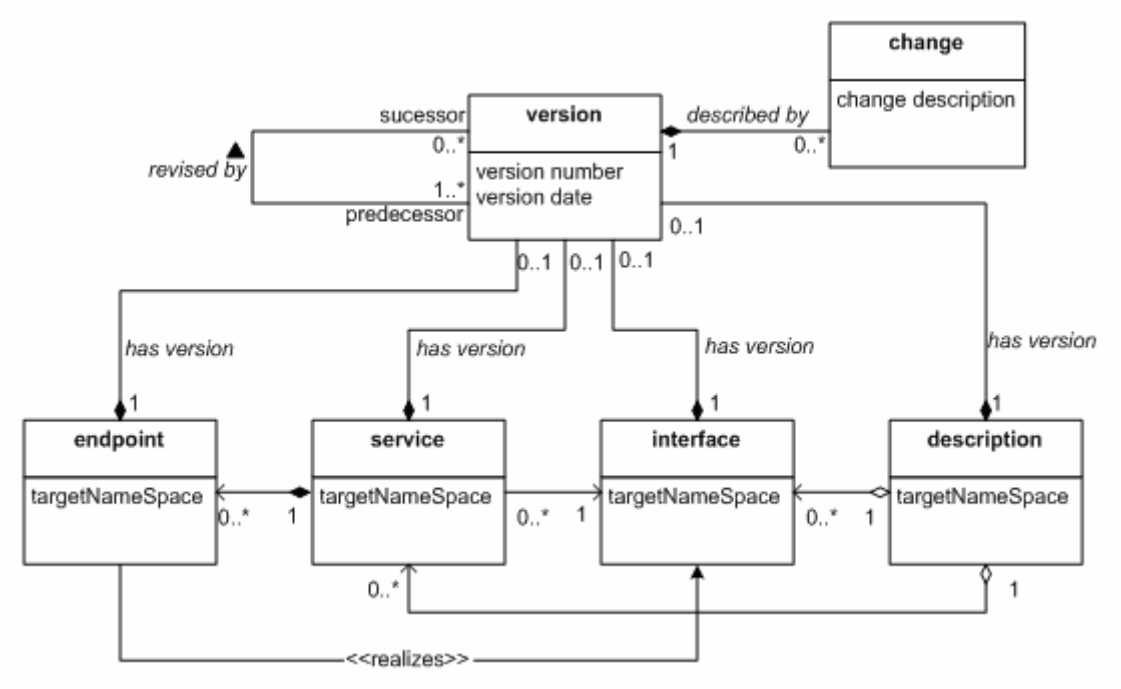

Figure 5. MOWS versioning concepts

Note that a set of consistent versions of each Web service element can be grouped into a revision. The idea of revision tagging Web services will be explored at a later time.

3 Web service endpoint manageability capabilities

The following sections define various manageability capabilities of a Web service endpoint.

Each capability is formally expressed in a UML diagram using the approach described in the MUWS specification, Section 4.

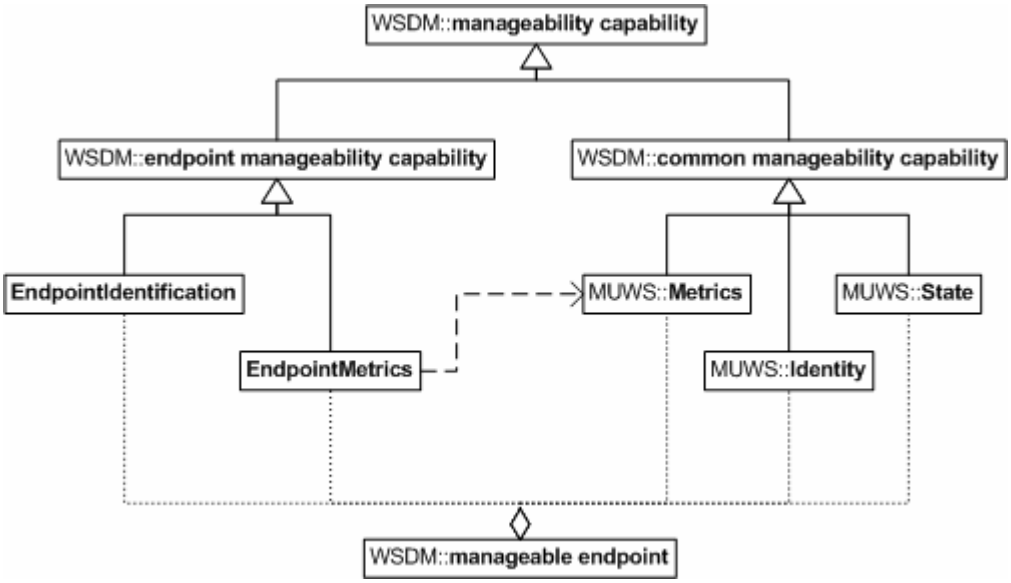

Figure 6. MOWS manageability capabilities conceptual taxonomy

Figure 6 depicts the conceptual taxonomy of MUWS and MOWS manageability capabilities. UML generalizations on the diagram are conceptual generalizations. For example, the MOWS **EndpointMetrics** "is a" WSDM endpoint manageability capability which "is a" WSDM manageability capability. The relationships between individual capability definitions are shown as UML dependencies. For example, the definition of the MOWS **EndpointMetrics** extends the definition of the MUWS **Metrics** capability.

Instances (implementations, realizations) of the individual manageability capabilities are then composed into an instance of the WSDM manageable endpoint concept. Such an instance would be an actual Web service endpoint whose implementation supports the composed capabilities.

The definitions (models) of the manageability capabilities of a Web service endpoint are rendered into WSDL elements (interfaces/portTypes) and supporting XML Schemas in Appendix D and Appendix E.

Following namespace prefixes are used in this document when referring to XML elements and XML schemas. The table below describes what prefix corresponds to which namespace URI.

Prefix	Namespace
muws-xs	http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/schema
muws-wsdl	http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/wsdl
mows-xs	http://docs.oasis-open.org/wsdm/2004/04/mows-0.5/schema
mows-wsdl	http://docs.oasis-open.org/wsdm/2004/04/mows-0.5/wsdl
wsa	http://schemas.xmlsoap.org/ws/2003/03/addressing
wsdl	http://www.w3.org/2002/07/wsdl
soap	http://www.w3.org/2002/12/soap-envelope
xs	http://www.w3.org/2001/XMLSchema

Unless otherwise specified, XML elements and XML schema types introduced below belong to the **mows-xs** namespace.

3.1 Identity

A WSDM manageable endpoint MUST support the MUWS **Identity** manageability capability. There are no extensions for the Web services endpoints defined or required for this capability.

3.2 Identification

The Web service endpoint's manageable identification capability is represented in the **EndpointIdentification** UML model class. The name of the class identifies the semantics of this capability. Note that this capability's name and semantics are consistent with the following definition (from the Webster dictionary).

> identification: **1 a :** an act of identifying **:** the state of being identified **b :** evidence of identity

This capability additionally provides the MUWS **Identity** capability's semantics, which are consistent with the following definition (from the Webster dictionary).

> identity: **1 a :** sameness of essential or generic character in different instances **b :** sameness in all that constitutes the objective reality of a thing **:** ONENESS

The *identification* capability is used to help establish the Web service endpoint being managed. The *identity* capability may be used to determine if two manageability providers manage the same resource or not.

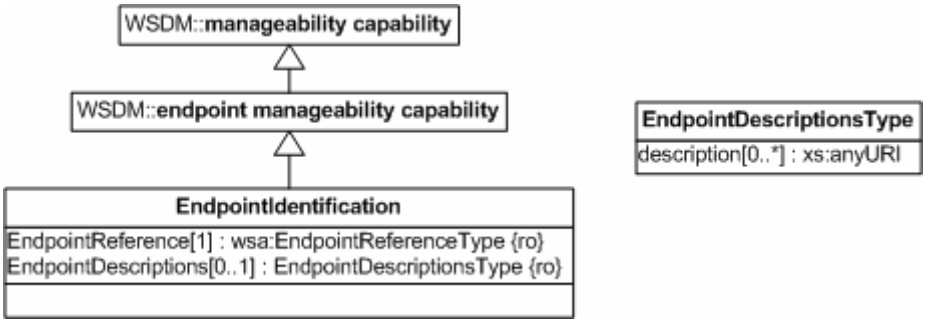

Figure 7. Endpoint identification manageability capability model

3.2.1 Properties

The following is the specification of the Web service endpoint identification properties (elements).

<**EndpointReference**>*wsa:EndpointReferenceType*</EndpointReference>
<**EndpointDescriptions**><description>*xs:anyURI*</description>*</EndpointDescriptions>?

EndpointReference is a reference to the Web service endpoint being managed. A reference must be resolvable to the actual useable endpoint. This property represents one way to access the endpoint resource but doesn't preclude the existence of multiple descriptions of the same endpoint resource.

EndpointDescriptions is a list of URIs pointing to description documents of the Web service endpoint resource. The different description documents can be of the same or of different types (e.g. WSDL1.1, WSDL2.0, UDDI tModel, etc.)

3.3 Metrics

The Web service endpoint's manageable metrics capability is represented in the **EndpointMetrics** UML model class. The name of the class identifies the semantics of this capability.

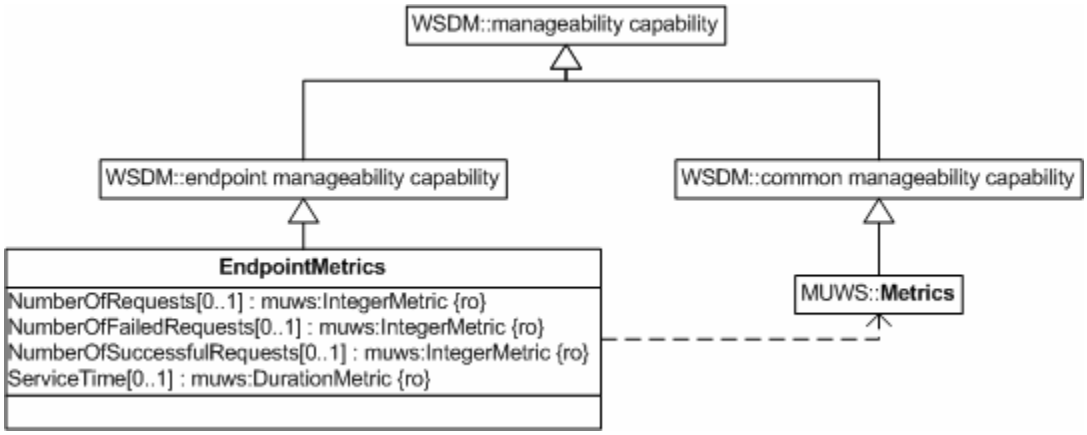

Figure 8. Endpoint metrics manageability capability model

This capability extends the definition of the MUWS Metrics capability. WSDM manageable endpoints that intend to support the **EndpointMetrics** capability MUST support the MUWS **Metrics** capability as well.

It is recommended that for adequate calculations, the Web service endpoint metric properties (one or all) are retrieved together with the **muws-xs:CurrentTime** property (e.g., using one request to retrieve multiple properties).

Metrics and request processing states are related. The request processing state change boundaries are the points where metric counters are incremented **[WSLC]**.

3.3.1 Properties

The following is the specification of the Web service endpoint metrics properties (elements).

```
<NumberOfRequests
        muws-xs:ChangeType="Counter">muws-xs:IntegerMetric</NumberOfRequests>?
<NumberOfFailedRequests
        muws-xs:ChangeType="Counter">muws-xs:IntegerMetric</NumberOfFailedRequests>?
<NumberOfSuccessfulRequests
        muws-xs:ChangeType="Counter"
        >muws-xs:IntegerMetric</NumberOfSuccessfulRequests>?
<ServiceTime
        muws-xs:ChangeType="Counter">muws-xs:DurationMetric</ServiceTime>?
```

NumberOfRequests is a counter of the number of request messages that the Web service endpoint has received.

NumberOfFailedRequests is a counter of the number of request messages that the Web service endpoint has received, and a (SOAP) fault was sent in reply.

NumberOfSuccessfulRequests is a counter of the number of request messages that the Web service endpoint has received, and anything but a (SOAP) fault was sent in reply.

ServiceTime is a counter of the total elapsed time it has taken the Web service endpoint to process all requests (successfully or not).

Note that **NumberOfSuccessfulRequests** + **NumberOfFailedRequests** ≤ **NumberOfRequests** as there could possibly be some requests that were received, but lost.

3.4 State

WSDM manageable endpoints that intend to support state management capability MUST support the MUWS **State** manageability capability. There are no extensions for the Web services endpoints defined or required for this capability.

The Web service lifecycle (WSLC) states defined by the W3C Web Services Architecture Management Task Force **[WSLC]** map to the MUWS states as follows:

- The WSLC **UP** state maps to the MUWS **Available** state. Any sub-state of WSLC **UP** MUST be mapped as MUWS **Available**.

460 - The WSLC **DOWN** state maps to the MUWS **Unavailable** state. Any sub-state of WSLC
461 **DOWN** SHOULD be mapped as MUWS **Unavailable**.
462 - The WSLC **SATURATED** sub-state of **DOWN** may be interpreted as the MUWS
463 **Degraded** state.

4 Example

This section is an example of a functional Web service for which a manageability endpoint exists. The example shows how to assemble MUWS and MOWS specification fragments to provide a manageability Web service. WSDL documents and SOAP messages are described.

Consider a description of a fictitious Web service – a mountain weather station. The following WSDL 1.1 document may, for example, be available at the http://weather.everest.org/service.wsdl URL.

```
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
             xmlns:s="http://www.w3.org/2001/XMLSchema"
             xmlns:s0="http://everest.org/"
             targetNamespace="http://everest.org/">
  <types>
    <s:schema elementFormDefault="qualified"
                      targetNamespace="http://everest.org/">
      <s:element name="GetCurrentTemperature">
        <s:complexType>
          <s:sequence>
            <s:element        name="altitude" type="s:double"
                              minOccurs="1" maxOccurs="1"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetCurrentTemperatureResponse">
        <s:complexType>
          <s:sequence>
            <s:element        name="GetCurrentTemperatureResult" type="s:double"
                              minOccurs="1" maxOccurs="1"/>
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </types>
  <message name="GetCurrentTemperatureSoapIn">
    <part name="parameters" element="s0:GetCurrentTemperature" />
  </message>
  <message name="GetCurrentTemperatureSoapOut">
    <part name="parameters" element="s0:GetCurrentTemperatureResponse" />
  </message>
  <portType name="WeatherStationSoap">
    <operation name="GetCurrentTemperature">
      <input message="s0:GetCurrentTemperatureSoapIn" />
      <output message="s0:GetCurrentTemperatureSoapOut" />
    </operation>
  </portType>
  <binding name="WeatherStationSoap" type="s0:WeatherStationSoap">
```

```
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
    <operation name="GetCurrentTemperature">
      <soap:operation style="document" />
      <input>
        <soap:body use="literal" />
      </input>
      <output>
        <soap:body use="literal" />
      </output>
    </operation>
  </binding>
  <service name="WeatherStation">
    <port name="WeatherStationSoap" binding="s0:WeatherStationSoap">
      <soap:address location="http://weather.everest.org/service"/>
    </port>
  </service>
</definitions>
```

The functional service, the weather station service, takes requests for a current temperature at a given altitude.

A manageability endpoint may exist that can let the weather station service be managed remotely. The following WSDL 1.1 document describes a WSDM-compliant manageability endpoint. The document may be available at the http://weather.everest.org/manageability.wsdl URL.

```
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:xs="http://www.w3.org/2001/XMLSchema"
             xmlns:wsrp="http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
             xmlns:muws-xs="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/schema"
             xmlns:mows-xs="http://docs.oasis-open.org/wsdm/2004/04/mows-0.5/schema"
             xmlns:muws-wsdl="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/wsdl"
             xmlns:mows-wsdl="http://docs.oasis-open.org/wsdm/2004/04/mows-0.5/wsdl"
             xmlns:s0="http://everest.org/"
             targetNamespace="http://everest.org/">

  <import namespace="http://www.ibm.com/xmlns/stdwip/web-services/WS-
ResourceProperties"
   location="http://www-106.ibm.com/developerworks/webservices/library/ws-resource/WS-
ResourceProperties.wsdl"/>
```

This imports definitions from the WS-ResourceProperties WSDL.

```
  <import namespace="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/wsdl"
location="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/wsdl"/>
  <import namespace="http://docs.oasis-open.org/wsdm/2004/04/mows-0.5/wsdl"
location="http://docs.oasis-open.org/wsdm/2004/04/mows-0.5/wsdl"/>
```

This imports WSDL definitons from the **muws-wsdl** and **mows-wsdl** namespaces.

```
  <types>
    <xs:schema elementFormDefault="qualified"
```

```
                targetNamespace="http://everest.org/">

    <xs:import namespace="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/schema"
schemaLocation="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/schema"/>
    <xs:import namespace="http://docs.oasis-open.org/wsdm/2004/04/mows-0.5/schema"
schemaLocation="http://docs.oasis-open.org/wsdm/2004/04/mows-0.5/schema"/>
```

This imports schema for **muws-xs** and **mows-xs** namespaces.

```
    <xs:complexType name="WeatherStationManageabilityPropertiesType">
                <xs:sequence>
                <xs:element ref="muws-xs:ResourceId"/>
                <xs:element ref="muws-xs:Name" minOccurs="0"/>
                <xs:element ref="muws-xs:Version" minOccurs="0"/>
                <xs:element ref="muws-xs:ResourceState"/>
                <xs:element ref="muws-xs:CurrentTime"/>
                <xs:element ref="mows-xs:EndpointReference"/>
                <xs:element ref="mows-xs:EndpointDescriptions" minOccurs="0"/>
                <xs:element ref="mows-xs:NumberOfRequests" minOccurs="0"/>
                <xs:element ref="mows-xs:NumberOfFailedRequests" minOccurs="0"/>
                <xs:element ref="mows-xs:NumberOfSuccessfulRequests" minOccurs="0"/>
                <xs:element ref="mows-xs:ServiceTime" minOccurs="0"/>
                <xs:any minOccurs="0" maxOccurs="unbounded"
                        namespace="##other" processContents="lax"/>
                </xs:sequence>
    </xs:complexType>
```

This type declares a property container for the weather station manageability endpoint.

```
    <xs:element name="WeatherStationManageabilityProperties"
                type="s0:WeatherStationManageabilityPropertiesType"/>
```

This element is the property container for the weather station manageability endpoint.

```
   </xs:schema>
  </types>
```

The following is the declaration of the interface (portType) of the weather station manageability endpoint.

```
  <portType name="WeatherStationManageabilitySoap"
          wsrp:ResourceProperties="s0:WeatherStationManageabilityProperties">
```

The **wsrp:ResourceProperties** points to the qualified name of the property container element.

The **GetResourceProperty** and **GetMultipleResourceProperties** operations belong to the WS-ResourceProperties specification and are directly mixed into this interface. Note that actual messages are declared in the **wsrp** namespace.

```
    <operation name="GetResourceProperty">
      <input  name="GetResourcePropertyRequest"
              message="wsrp:GetResourcePropertyRequest" />
      <output name="GetResourcePropertyResponse"
              message="wsrp:GetResourcePropertyResponse" />
      <fault  name="UnknownResource"
              message="wsrp:ErrorMessage" />
```

```
    <fault  name="InvalidResourcePropertyQName"
            message="wsrp:ErrorMessage" />
  </operation>
  <operation name="GetMultipleResourceProperties">
    <input  name="GetMultipleResourcePropertiesRequest"
            message="wsrp:GetMultipleResourcePropertiesRequest" />
    <output name="GetMultipleResourcePropertiesResponse"
            message="wsrp:GetMultipleResourcePropertiesResponse" />
    <fault  name="UnknownResource"
            message="wsrp:ErrorMessage" />
    <fault  name="InvalidResourcePropertyQName"
            message="wsrp:ErrorMessage" />
  </operation>
```

The **Start**, **Stop** and **ResetAll** operations belong to the MUWS specification and are directly mixed into this interface. Note that actual messages are declared in the **muws-wsdl** namespace.

```
  <operation name="Start">
    <input name="StartRequest" message="muws-wsdl:StartRequest"/>
    <output name="StartResponse" message="muws-wsdl:StartResponse"/>
  </operation>
  <operation name="Stop">
    <input name="StopRequest" message="muws-wsdl:StopRequest"/>
    <output name="StopResponse" message="muws-wsdl:StopResponse"/>
  </operation>
  <operation name="ResetAll">
    <input name="ResetAllRequest" message="muws-wsdl:ResetAllRequest"/>
    <output name="ResetAllResponse" message="muws-wsdl:ResetAllResponse"/>
  </operation>
 </portType>
```

The following is the SOAP document/literal binding of the interface declared above.

```
 <binding name="WeatherStationManageabilitySoap"
type="s0:WeatherStationManageabilitySoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
  <operation name="GetResourceProperty">
    <soap:operation style="document" />
    <input>
      <soap:body use="literal" />
    </input>
    <output>
      <soap:body use="literal" />
    </output>
  </operation>
  <operation name="GetMultipleResourceProperties">
    <soap:operation style="document" />
    <input>
      <soap:body use="literal" />
    </input>
    <output>
      <soap:body use="literal" />
    </output>
  </operation>
  <operation name="Start">
    <soap:operation style="document" />
```

```
    <input>
      <soap:body use="literal" />
    </input>
    <output>
      <soap:body use="literal" />
    </output>
  </operation>
  <operation name="Stop">
    <soap:operation style="document" />
    <input>
      <soap:body use="literal" />
    </input>
    <output>
      <soap:body use="literal" />
    </output>
  </operation>
  <operation name="ResetAll">
    <soap:operation style="document" />
    <input>
      <soap:body use="literal" />
    </input>
    <output>
      <soap:body use="literal" />
    </output>
  </operation>
 </binding>
```

The following is the description of the manageability service which contains the weather station manageability endpoint.

```
 <service name="WeatherStationManageability">
  <port name="WeatherStationManageabilitySoap"
binding="s0:WeatherStationManageabilitySoap">
    <soap:address location="http://weather.everest.org/manageability"/>
  </port>
 </service>
</definitions>
```

According to the description of the weather station manageability endpoint, one may retrieve Web service endpoint metrics. Metrics are about the functional Web service, in this case the weather station service, but their request is sent to the manageability endpoint. For example, to retrieve the number of requests received by the weather station Web service endpoint, one may send the following SOAP message to the http://weather.everest.org/manageability URL via the HTTP protocol.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<GetResourcePropertyRequest xmlns:q1="http://docs.oasis-open.org/wsdm/2004/04/mows-
0.5/schema" xmlns="http://www.ibm.com/xmlns/stdwip/web-services/WS-
ResourceProperties">q1:NumberOfRequests</GetResourcePropertyRequest>
</soap:Body>
```

```
</soap:Envelope>
```

The response from the weather station manageability endpoint to the above request may be the following SOAP message.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<GetResourcePropertyResponse xmlns="http://www.ibm.com/xmlns/stdwip/web-services/WS-
ResourceProperties">
<mows-xs:NumberOfRequests xmlns:mows-xs="http://docs.oasis-
open.org/wsdm/2004/04/mows-0.5/schema">130</mows-xs:NumberOfRequests>
</GetResourcePropertyResponse>
</soap:Body>
</soap:Envelope>
```

5 References

5.1 Normative

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

[MUWS] http://docs.oasis-open.org/wsdm/2004/04/muws-0.5

[WSA] http://xml.coverpages.org/WS-Addressing20030523-IBM.pdf

[WSRP] http://www.ibm.com/developerworks/library/ws-resource/ws-resourceproperties.pdf

[WSDL] http://www.w3.org/TR/wsdl

[SOAP] http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[XMLS] http://www.w3.org/TR/xmlschema-1/, http://www.w3.org/TR/xmlschema-2/

[XML] http://www.w3.org/TR/REC-xml

[XNS] http://www.w3.org/TR/REC-xml-names/

5.2 Non-normative

[Charter] http://www.oasis-open.org/committees/wsdm/charter.php

[MOWS-Reqs] http://www.oasis-open.org/apps/org/workgroup/wsdm/download.php/3887/WSDM-MOWS-Requirements.20031008.doc

[WS-Arch] http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

[WSLC] http://www.w3.org/TR/2004/NOTE-wslc-20040211/

Appendix A. Acknowledgments

The following people made contributions to this specification:

- Brian Carol <brian.carroll@merant.com>
- Fred Carter <fred.carter@amberpoint.com>
- John DeCarlo <jdecarlo@mitre.org>
- Andreas Dharmawan <andreas@westbridgetech.com>
- Heather Kreger <kreger@us.ibm.com>
- Bryan Murray <bryan.murray@hp.com>
- Micheal Perks <mperks@us.ibm.com>
- Igor Sedukhin <Igor.Sedukhin@ca.com>
- William Vambenepe <vbp@hp.com>
- Andrea Westerinen <andreaw@cisco.com>

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

- Guru Bhat <Guru.Bhat@oracle.com>
- Jeff Bohren <jbohren@opennetwork.com>
- Winston Bumpus <winston_bumpus@dell.com>
- Fred Carter <fred.carter@amberpoint.com>
- John DeCarlo <jdecarlo@mitre.org>
- Andreas Dharmawan <andreas@westbridgetech.com>
- Mark Ellison <ellison@ieee.org>
- Daniel Foody <dan@actional.com>
- Heather Kreger <kreger@us.ibm.com>
- Bryan Murray <bryan.murray@hp.com>
- Paul Lipton <paul.lipton@ca.com>
- Hal Lockhart <hlockhar@bea.com>
- Rajiv Maheshwari <rajiv.k.maheshwari@oracle.com>
- Richard Nikula <Richard_Nikula@bmc.com>
- Michael Perks <mperks@us.ibm.com>
- Homayoun Pourheidari <homayoun@hp.com>
- Karl Schopmeyer <k.schopmeyer@attglobal.net>
- Igor Sedukhin <Igor.Sedukhin@ca.com>
- Davanum Srinivas <Davanum.Srinivas@ca.com>
- Ellen Stokes <stokese@us.ibm.com>
- Thomas Studwell <studwell@us.ibm.com>
- Ryoichi Ueda <ueda@sdl.hitachi.co.jp>
- William Vambenepe <vbp@hp.com>

803 • Andrea Westerinen <andreaw@cisco.com>

Appendix B. Revision History

Rev	Date	By Whom	What
wd-01	2003-10-31	Igor Sedukhin	Initial version & content
wd-02	2003-11-14	Igor Sedukhin	Versioning content, Identification model content, fixes from e-mail and phone discussions.
wd-03	2003-12-02	Igor Sedukhin	Updated identification model, added configuration model. Fixed MOWS locus of implementation diagram.
wd-04	2004-01-26	Igor Sedukhin	Changes pending from F2F and e-mail discussions.
wd-05	2004-02-17	Igor Sedukhin	Added Metrics capability specification. Modified Identification capability specification to include XML fragments. Fixed the document in other places (editorial).
wd-06	2004-03-01	Igor Sedukhin	Replaced versioning with the text and diagram from Mike Perks. Added appendix with web service lifecycle from Heather Kreger. Added preliminary text in the example section 4. Fixed metrics UML model and added text explaining the dependency on MUWS. Fixed identification UML model to match XML Schema element declaration. Added normative WSDL and XML Schema in appendices D and E.
wd-07	2004-03-18	Igor Sedukhin	Added abstract, XNS reference, Charter reference, mission statement in section 1. Aligned Terminology with MUWS. Reworded “specification of the above…” in section 2. Aligned QName examples with XNS spec. Moved UML template text to MUWS. Inserted UML conceptual taxonomy diagrams (aggregated MOWS capabilities diagram). Added Identity and State section to mimic MUWS “profile”. Added statement that MOWS Metrics extends MUWS Metrics and a reference to Appendix F.
wd-08	2004-03-19	Igor Sedukhin	Separated namespaces of schemas and WSDLs.
wd-09	2004-03-24	Igor Sedukhin	Fixed model names. Fixed namespaces of MOWS/MUWS schema and WSDL. Fixed references and links. Removed

Rev	Date	By Whom	What
			xsd:. Fixed import locations in the example section. Pasted proper WSDL and schema in the appendices.
wd-10	2004-03-24	Igor Sedukhin	Fixed namespaces, optionality of some properties, added Composability section, naming conventions section and pasted latest WSDL and schema. Added requirements reference.
cd	2004-04-02	Igor Sedukhin	Fixed ResourceId, ##other & lax. Made all metrics properties optional. MOWS state mapped to WSLC and removed the appendix F.

805

Appendix C. 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.

Appendix D. XML Schemas

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
        xmlns:muws-xs="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/schema"
        xmlns:mows-xs="http://docs.oasis-open.org/wsdm/2004/04/mows-0.5/schema"
        targetNamespace="http://docs.oasis-open.org/wsdm/2004/04/mows-0.5/schema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">

<xs:import namespace="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/schema"
        schemaLocation="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/schema"/>
<xs:import namespace="http://schemas.xmlsoap.org/ws/2003/03/addressing"
        schemaLocation="http://schemas.xmlsoap.org/ws/2003/03/addressing"/>

<xs:element name="EndpointReference" type="wsa:EndpointReferenceType"/>
<xs:element name="EndpointDescriptions">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="description" type="xs:anyURI"
                                minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>

<xs:element name="NumberOfRequests" type="muws-xs:IntegerMetric"/>
<xs:element name="NumberOfSuccessfulRequests" type="muws-xs:IntegerMetric"/>
<xs:element name="NumberOfFailedRequests" type="muws-xs:IntegerMetric"/>
<xs:element name="ServiceTime" type="muws-xs:DurationMetric"/>

<xs:complexType name="EndpointIdentificationPropertiesType">
        <xs:sequence>
                <xs:element ref="mows-xs:EndpointReference"/>
                <xs:element ref="mows-xs:EndpointDescriptions" minOccurs="0"/>
                <xs:any minOccurs="0" maxOccurs="unbounded"
                        namespace="##other" processContents="lax"/>
        </xs:sequence>
</xs:complexType>

<xs:element name="EndpointIdentificationProperties"
        type="mows-xs:EndpointIdentificationPropertiesType"/>

<xs:complexType name="EndpointMetricsPropertiesType">
        <xs:sequence>
                <xs:element ref="mows-xs:NumberOfRequests" minOccurs="0"/>
                <xs:element ref="mows-xs:NumberOfFailedRequests" minOccurs="0"/>
                <xs:element ref="mows-xs:NumberOfSuccessfulRequests" minOccurs="0"/>
                <xs:element ref="mows-xs:ServiceTime" minOccurs="0"/>
                <xs:any minOccurs="0" maxOccurs="unbounded"
                        namespace="##other" processContents="lax"/>
        </xs:sequence>
</xs:complexType>

```

```
<xs:element name="EndpointMetricsProperties"
        type="mows-xs:EndpointMetricsPropertiesType"/>

</xs:schema>
```

Appendix E. WSDL elements

```
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:wsrp="http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
        xmlns:muws-xs="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/schema"
        xmlns:mows-xs="http://docs.oasis-open.org/wsdm/2004/04/mows-0.5/schema"
        xmlns:muws-wsdl="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/wsdl"
        xmlns:mows-wsdl="http://docs.oasis-open.org/wsdm/2004/04/mows-0.5/wsdl"
        targetNamespace="http://docs.oasis-open.org/wsdm/2004/04/mows-0.5/wsdl">

  <types>
  <xs:schema elementFormDefault="qualified"
        targetNamespace="http://docs.oasis-open.org/wsdm/2004/04/mows-0.5/wsdl">

        <xs:import namespace="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/schema"
                schemaLocation="http://docs.oasis-open.org/wsdm/2004/04/muws-
0.5/schema"/>
        <xs:import namespace="http://docs.oasis-open.org/wsdm/2004/04/mows-0.5/schema"
                schemaLocation="http://docs.oasis-open.org/wsdm/2004/04/mows-
0.5/schema"/>

  </xs:schema>
  </types>

  <portType name="EndpointIdentification"
        wsrp:ResourceProperties="mows-xs:EndpointIdentificationProperties"/>

  <portType name="EndpointMetrics"
        wsrp:ResourceProperties="mows-xs:EndpointMetricsProperties"/>
</definitions>
```