



1

2

3

4

# Web Services Distributed Management: Management Using Web Services (MUWS 0.5)

5

## Committee Draft, 2 April 2004

6

### Document identifier:

7

cd-wsdm-muws-0.5

8

### Location:

9

<http://docs.oasis-open.org/wsdm/2004/04/muws-0.5>

10

### Editors:

11

*Andreas Dharmawan, Westbridge Technology* <[andreas@westbridgetech.com](mailto:andreas@westbridgetech.com)>

12

*William Vambenepe, Hewlett-Packard* <[william\\_vambenepe@hp.com](mailto:william_vambenepe@hp.com)>

13

### Abstract:

14

There are two parts of Web services Distributed Management: Management *Using* Web services and Management *of* Web services. This specification defines the former.

15

16

Management *Using* Web services defines how an Information Technology resource connected to a network provides the manageability interfaces such that it can be managed remotely using Web services technologies.

17

18

19

### Status:

20

This document is a committee draft approved by the WSDM TC. It is not intended to become an OASIS standard. There is no guarantee that any part of its content will appear in the final release specification, MUWS 1.0.

21

22

23

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

24

25

26

27

28

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/>).

29

30

31

32

The errata document for this specification is at:

33

<http://docs.oasis-open.org/wsdm/2004/04/muws-0.5-errata>

34

---

## Table of Contents

36	1	Introduction.....	4
37	1.1	Terminology .....	5
38	1.2	Notational conventions .....	5
39	2	Architecture .....	6
40	2.1	Context .....	6
41	2.2	Conceptual Model .....	7
42	2.3	Logical Model.....	8
43	2.3.1	Role Definitions.....	9
44	2.4	Composability .....	10
45	2.5	Processing Model.....	11
46	2.5.1	Prerequisites.....	11
47	2.5.2	Discovery.....	11
48	2.5.3	Interaction.....	12
49	3	Support from Web Services Platform .....	13
50	4	Manageability Capabilities .....	14
51	4.1	Identity .....	15
52	4.1.1	Definition.....	15
53	4.1.2	Data Types.....	16
54	4.1.3	Properties .....	16
55	4.2	State .....	18
56	4.2.1	Definition.....	18
57	4.2.2	Description of State Model .....	18
58	4.2.3	Data Types.....	19
59	4.2.4	Properties .....	20
60	4.2.5	Operations .....	20
61	4.3	Metrics .....	21
62	4.3.1	Definition.....	21
63	4.3.2	Data Types.....	22
64	4.3.3	Properties .....	23
65	4.3.4	Operations .....	24
66	5	Discovery and Introspection.....	25
67	6	Defining a Manageability Interface.....	26
68	7	References.....	27

69	7.1	Normative .....	27
70	7.2	Non-normative .....	28
71	Appendix A.	Acknowledgements .....	29
72	Appendix B.	Notices.....	31
73	Appendix C.	Schemas (Normative) .....	32
74	Appendix D.	WSDL elements (Normative) .....	35
75	Appendix E.	Web Services Platform .....	37
76	Initial Focus .....		37
77	Properties.....		37
78	Meta Data.....		37
79	Addressing.....		38
80	Notification .....		38
81	Versioning .....		38
82	Security .....		39
83	Registration and Discovery .....		39
84	Future Focus .....		39
85	Policy .....		40
86	Name Resolution.....		40
87	Transaction.....		40
88	Flow .....		41
89	Negotiation .....		41
90			

91

# 1 Introduction

92

**Management Using Web Services (MUWS)** enables management of distributed IT resources using Web services. Many distributed IT resources use different management interfaces. By leveraging Web service technology, MUWS enables easier and more efficient IT management systems by providing a flexible common framework for manageability interfaces that benefits from the features of Web services protocols. Universal management interoperability across the many different varieties of distributed IT resources can be achieved using MUWS.

98

The types of management capabilities exposed by MUWS are the management capabilities generally expected in distributed IT management systems. Examples of manageability functions that can be performed via **MUWS** include:

99

101

- monitoring quality of services

102

- enforcing service level agreements

103

- controlling tasks

104

- managing resource life-cycles

105

106

MUWS is designed to meet the requirements defined in the MUWS Requirements document **[MUWS REQS]**. Whenever possible, MUWS leverages existing Web services specifications to ensure interoperability, adoptability, and extensibility.

107

109

There is a minimum set of manageability capabilities the manageability provider must support in order to participate in MUWS. This minimum set of manageability capabilities is defined in this specification.

110

111

112

Additionally, the methods and mechanisms provided by MUWS for discovering the manageability interfaces of manageable IT resources are discussed in this specification.

113

114

Finally, the manageability interface itself is defined in this specification.

115

To understand the various topics discussed in this specification, the reader should be familiar with the IT management concepts. In addition, the following assumptions are made:

116

117

- The reader is familiar with the Web Services Architecture **[WSA]**

118

- The reader is familiar with XML **[XML1.0 3<sup>rd</sup> Edition]**, XML Schema **[XML Schema Part 1]** **[XML Schema Part 2]**, and XML Namespace **[XNS]**

119

120

- The reader is familiar with WSDL **[WSDL1.1]**, SOAP **[SOAP1.1]**, UDDI **[UDDI]**

121

- The reader is familiar with WS-ResourceProperties **[WS-ResourceProperties]**, WS-Addressing **[WS-Addressing]**

122

123

Section 3, 4, 5, 6 and appendices C (schemas) and D (WSDL elements) are normative specifications with the following **exception**: UML illustrations found in section 4 are non-normative. The rest of the document is a non-normative, explanatory material intended to ease the understanding.

124

125

126

## 127 **1.1 Terminology**

128 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD",  
129 "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be  
130 interpreted as described in RFC 2119 [RFC2119].

## 131 **1.2 Notational conventions**

132 This specification uses an informal syntax to describe the XML grammar of the messages  
133 making up the management interfaces. This syntax uses the following rules:

- 134 § The syntax appears as an XML instance, but data types appear instead of values.
- 135 § {any} is a placeholder for elements from some other namespace (like ##other in the XML  
136 Schema).
- 137 § The Cardinality of an attribute, element, or {any}, is indicated by appending characters to  
138 the item as follows:
- 139           ? none, or one
- 140           \* none, or more
- 141           + one, or more
- 142           No character exactly one
- 143 • Items contained within the square brackets, [ and ], are treated as a group.
- 144 § Items separated by | and grouped within parentheses, ( and ), indicate syntactic  
145 alternatives.
- 146 § Three consecutive periods, ... are used in XML start elements to indicate that attributes  
147 from some other namespace are allowed.
- 148 § The XML namespace prefixes, defined below, indicate the namespace of the Item.

149 When defining operations, this specification uses pseudo-schema to describe the input and, if  
150 appropriate, output messages. A full WSDL description of all operations is available in the  
151 appendix of this specification.

152

153

## 2 Architecture

154

### 2.1 Context

155  
156

This section provides a context for the MUWS Architecture. The MUWS Architecture makes use of the Web Services Architecture (WSA).

157  
158  
159  
160  
161  
162

WSA provides a common definition of a Web service, as well as a conceptual model and context for understanding Web services and the relationships between Web service components. WSA describes the characteristics common to all Web services and describes some characteristics needed by many, but not all, Web services. Additionally, by identifying the global elements of the global Web services network that are required to ensure interoperability among Web services, WSA provides an architecture of interoperability for MUWS

163  
164  
165  
166  
167

Since WSA defines how to specify information and operations through WSDL interfaces, access through bindings, and discovery through endpoints it is consistent to use WSA to describe, as well as provide access to, and discoverability of, the manageable components of the WSA itself. In fact, this paradigm can be extended by providing access to, and discoverability of, any manageable resource in the IT infrastructure.

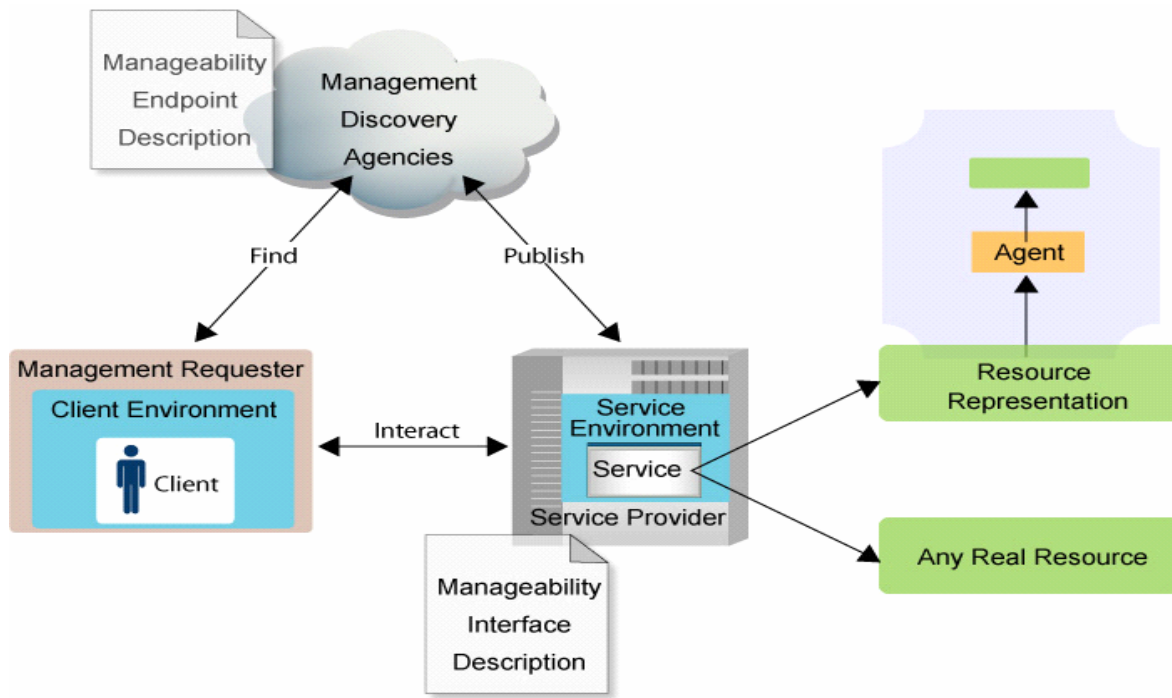
168  
169  
170

The management interface of a resource can be exposed through a Web service interface in the same way as the functional interface. This is true even if Web services are not used to provide access to the functionality of the resource.

171  
172

Figure 2.1-1 illustrates two possible methods a manageable Web service may use to fulfill management requests on an exposed resource.

173



174  
175

Figure 2.1-1, MUWS on IT Resource

176 With the first method, a manageability service interacts directly with an exposed resource  
177 through a supported management interface. For example, the manageability service may  
178 interact with an exposed resource supporting Java's JMX facility, or interact with an exposed  
179 resource supporting a proprietary C API.

180 With the second method, the manageability service interacts indirectly with an exposed resource  
181 through a management agent acting as an intermediary. This management agent interacts  
182 either directly or indirectly with the exposed resource.

183 This second method enables existing management instrumentation of a resource to be exported  
184 through Web services to a management consumer.

185 Whether the manageability service uses the direct method or the indirect method, the  
186 management consumer is provided a consistent view of the management instrumentation of the  
187 exposed resource. The management consumer remains unaware of any legacy management  
188 agent, facility or API utilized by the manageability service.

189 Using Web services to describe and provide access to a manageability interface for an exposed  
190 resource creates a consistent, cross platform, cross vendor, and potentially cross enterprise  
191 interaction model for consumers and producers of management information.

192 Creating a Web services to access manageability information of an exposed resource does not  
193 preclude alternate means to access this information. For example, a management consumer is  
194 able to use any and all of WSDM MUWS, SNMP or CIM/WBEM to access manageability  
195 information for an exposed resource.

## 196 **2.2 Conceptual Model**

197 This MUWS specification defines how manageability of an arbitrary IT resource can be accessed  
198 via Web services. Manageability is one possible quality of a resource. Manageability is  
199 composed of a number of capabilities. Each capability has its own distinct semantics. Therefore,  
200 a manageable resource composes a set of manageability capabilities. Figure 2.3-1 relates the  
201 concepts necessary for MUWS.

202 According to the concepts in the WSDL specification, a Web service is an aggregate of  
203 endpoints. Each endpoint offers the Web service at an address that is accessible according to a  
204 binding. A Web service has some number of interfaces that are realized by its endpoints.

205 Each interface describes a set of named messages, and their formats, that can be exchanged.  
206 Properly formatted messages can be sent to the address of an endpoint in a way prescribed by  
207 the binding. A description, either as a document, or an artifact, is composed with definitions of  
208 interfaces and services. A description may contain definitions of interfaces, or definitions of  
209 services, or definitions of both.

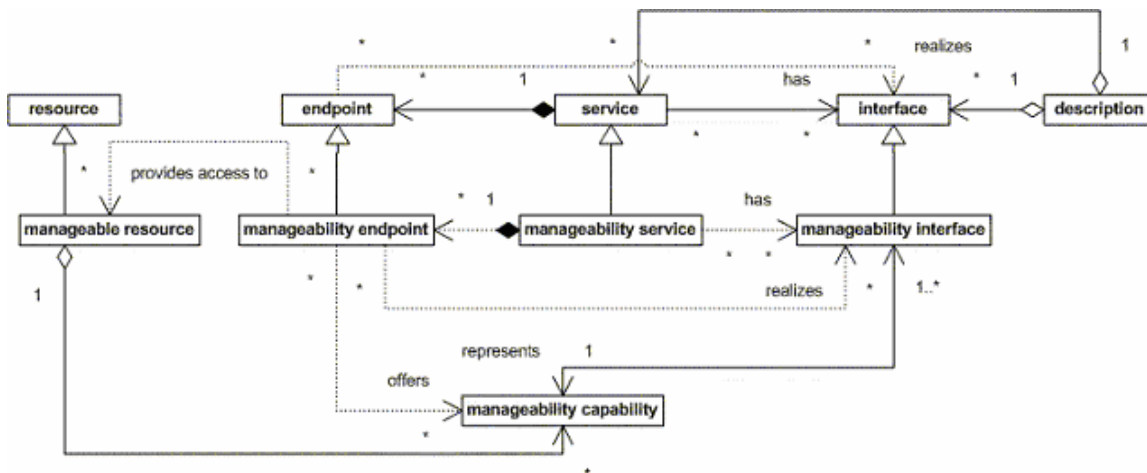
210 In accordance with the Web services concepts expressed above, access to the manageability for  
211 a resource must be provided by an endpoint. We call such an endpoint a manageability  
212 endpoint. Implicitly, a manageability endpoint belongs to a manageability service, which has a  
213 number of manageability interfaces that are realized by manageability endpoints.

214 Thus, a single manageability interface represents all or part of a manageability capability.  
215 Similarly, a single manageability capability may be represented by one or more interfaces. The  
216 semantics of a particular interface includes the description of the set of possible message  
217 exchanges. The exchanges are rendered as message formats grouped into one or more  
218 interfaces.

219 For example, the ability to offer metrics could be captured in a "Metrics" UML model which is an  
220 instance of the manageability capability concept. The semantics of offering metrics could be  
221 rendered from the UML model into a WSDL interface description defined within the

222 "http://docs.oasis-open.org/wsdm/2004/04/manageability/metrics" namespace. Such a rendering  
 223 would be an instance of the manageability interface concept.

224 This specification defines the base set of manageability capabilities that can be composed into a  
 225 manageable resource or combined into aggregate capabilities. For example, a  
 226 TotallyManageableResource uber-capability could be defined that includes all of the base  
 227 manageability capabilities defined in this specification. Such an aggregate manageability  
 228 capability could also be composed into a manageable resource, and in that sense, an aggregate  
 229 manageability capability is conceptually the same as any other capability. However, this  
 230 specification does not currently attempt to define or identify aggregate capabilities. Rather, this  
 231 specification focuses on the definition of the base set manageability capabilities.



232

Figure 2.3-1, MUWS Concepts

233

234

## 235 2.3 Logical Model

236 A manageability provider may provide manageability capabilities for many resources. In other  
 237 words a manageability provider may allow many resources to be exposed as manageable  
 238 resources.

239 To accomplish this, a manageability provider maintains manageability endpoints. A  
 240 manageability endpoint provides a means to access one or more manageable  
 241 resources. According to our conceptual definition, a manageable resource is a resource  
 242 composed with any number of manageability capabilities. In order to compose capabilities into  
 243 the manageable resource, a manageability provider supports the manageability capabilities  
 244 offered by its manageability endpoints.

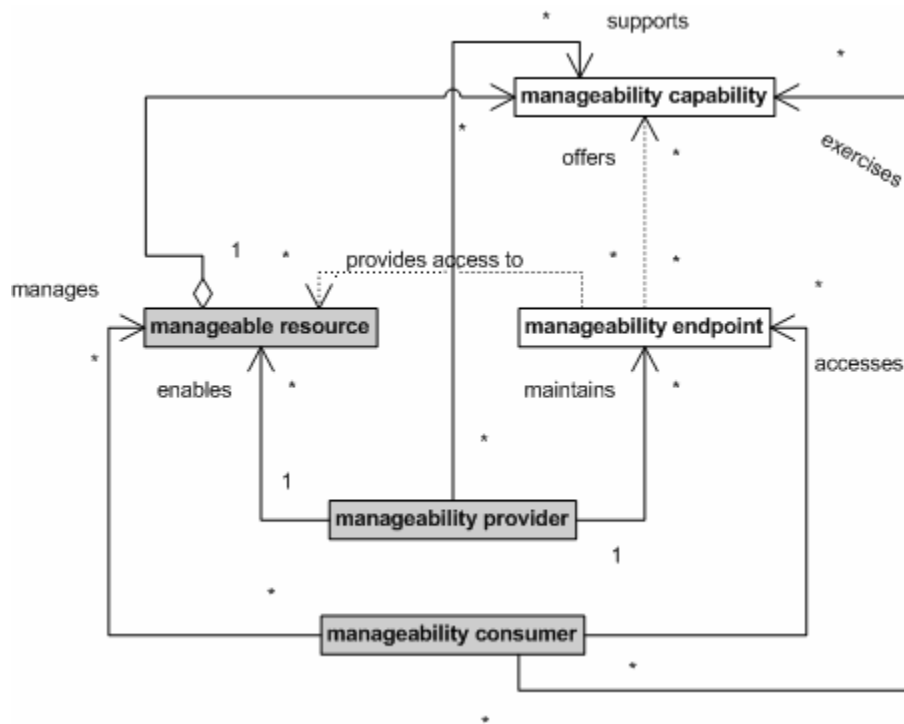
245 For example, a manageability provider could embed code into a resource supporting the  
 246 manageability capabilities of the resource, thus making the resource manageable. A  
 247 manageability provider could also support manageability capabilities by deploying resources  
 248 within a container that supports manageability capabilities for all its resources.

249 The manageability consumer manages manageable resources. To *manage* in this context means  
 250 to exert control and to obtain and interpret information about the manageable resource. In order  
 251 to manage the manageable resource, the consumer accesses manageability endpoints and  
 252 exercises the manageability capabilities offered by the endpoint.

253 To *exercise* in this context means to use the distinct semantics of a manageability capability.  
 254 Essentially, the consumer exercises an understanding of the semantics defined by a  
 255 manageability capability, but exercises this understanding on an actual manageable resource. In  
 256 a technical sense, to *exercise* offered manageability capabilities translates into using a distinct  
 257 group of properties, operations, events and metadata by exchanging messages with the  
 258 manageability endpoint.

259 For example, consider a server with several disk drives. The manageability provider could be a  
 260 software process running on the server. This manageability provider could allow each disk drive  
 261 to become a manageable resource by providing a manageability endpoint for each disk drive.  
 262 The implementation of these endpoints by the provider could use OS calls to access the disk  
 263 drives. This pattern could be used to provide manageability capabilities exposed through  
 264 manageability endpoints, such as retrieving the amount of disk space available in each disk  
 265 drive.

266 An IT management console, acting as a manageability consumer, could then exercise this  
 267 manageability property by connecting to the manageability endpoints provided by the  
 268 manageability provider. Using this example, the management console could retrieve the amount  
 269 of disk space available on each disk drive.



270  
 271

Figure 2.4-1, MUWS Logical Model

### 272 2.3.1 Role Definitions

273 This section defines the roles and interactions of the major components of MUWS, and related  
 274 components, within the MUWS Architecture. This section is not intended to constrain the locus  
 275 of implementation., Rather, this section documents the required components, their roles, and  
 276 how they interact.

277 NOTE: One implementation may be a single component with many roles while another  
 278 implementaion may be composed of several components, each with an assigned role.

279 The major roles are the manageability consumer, the manageability provider, and the  
280 manageability resource. These roles are represented by the shaded boxes in the MUWS Logical  
281 Model (Figure 2.4.-1).

### 282 **2.3.1.1 Manageability Consumer**

283 The manageability consumer:

- 284 • Consumes management information about the resource
- 285 • Manages, monitors, configures the resource
- 286 • Understands the manageability capabilities of the resource

### 287 **2.3.1.2 Manageability Provider**

288 The manageability provider:

- 289 • Provides the manageability quality for a resource, enabling a resource to become a  
290 manageable resource
- 291 • Provides management information for consumers according to the manageability  
292 capabilities of the resource

293 NOTE: The manageability provider may be implemented in the manageable resource or it may  
294 not. The manageability provider may provide the manageability quality for more than one  
295 resource. Thus, the role of manageability provider is not intended to constrain the locus of  
296 implementation.

### 297 **2.3.1.3 Manageable Resource**

298 The Manageable Resource is an IT resource that is manageable via a MUWS based  
299 infrastructure. Because there are no restrictions on the locus of implementation, the  
300 manageable resource may, or may not, implement the role of manageability provider for the  
301 manageability service.

302 A manageable resource does not have to be fine-grained. To illustrate this point, a complete  
303 server could be a manageable resource. The manageability provider offering the manageability  
304 endpoint for the server can run either on the server, or on some other machine. At an even  
305 higher level of granularity, a server farm could be exposed as a single manageable resource. For  
306 example, an IT management software package could act as a manageability provider offering  
307 manageability endpoints for the server farm, and exposing manageability capabilities. Examples  
308 of manageability capabilities might be retrieving the aggregate available disk space for all  
309 servers in the farm, or, bringing up or down all servers in the farm. Such a manageability  
310 provider for the server farm could be implemented using a set of proprietary interfaces for the  
311 servers in the farm., Alternatively, if the servers are manageable resources, such a  
312 manageability provider could act as a manageability consumer by accessing and aggregating the  
313 manageability capabilities of the servers in the farm.

## 314 **2.4 Composability**

315 MUWS allows the resource and its service to be manageable in a standard and interoperable  
316 manner. This is achieved by defining the manageability capabilities and interfaces of a  
317 resource. A resource may support both manageability capabilities and functional capabilities. In  
318 this case, the resource may allow manageability consumers access to appropriate functional

319 capabilities along with the manageability capabilities. Managers could discover such composition  
320 by inspecting the service description.

321 Managers could take advantage of the composition of manageability with functional capabilities  
322 by querying for free disk space using a disk manageability capability and then reading disk  
323 sectors using a disk functional capability. Composability makes it easy for implementers of a  
324 resource service to offer an appropriate subset of functional capabilities along with its  
325 manageability capabilities.

## 326 **2.5 Processing Model**

327 Compliant implementations of the roles as defined in section 2.3, the logical model, act  
328 according to the following basic processing rules.

### 329 **2.5.1 Prerequisites**

330 A manageability consumer and a manageability provider must understand the information model  
331 within which semantics of a manageability capability are described. For example, a UML model  
332 could express a group of properties, operations, events and metadata. The meaning of what the  
333 model defines must be equally understood by both parties, provider and consumer. The base  
334 capabilities defined in MUWS, as well as the capabilities defined in domain-specific  
335 specifications built on top of MUWS, are the means to achieving such an understanding.

336 A manageability consumer and a manageability provider both must understand how to identify  
337 which manageability interface corresponds to with a manageability capability, and vice versa.  
338 The base capabilities described in MUWS are the means to achieving such and understanding.

339 A manageability consumer must be able to obtain the description of the manageability service,  
340 its endpoints and necessary manageability interfaces. The manageability provider, or another  
341 party on behalf of the provider, makes such descriptions available to the consumer.

342 A manageability provider must be able to obtain the description of the manageability interfaces  
343 for the capabilities it wants to support. MUWS includes in-line descriptions, usually as  
344 appendices, of XML Schemas and WSDL elements for manageability capabilities. MUWS also  
345 indicates URL locations of such XML schema documents and WSDL documents hosted on the  
346 OASIS Web site.

### 347 **2.5.2 Discovery**

348 A manageability consumer discovers the necessary manageable resources by discovering the  
349 manageable endpoints, reading their descriptions and exchanging messages as required. MUWS  
350 indicates which manageability capabilities can be used to discover other Web services endpoints  
351 or other manageability endpoints.

352 To discover manageable endpoints, the consumer uses the same discovery techniques as used  
353 for any Web service endpoints. For example, the consumer may be provided with a URL  
354 referencing a WSDL document describing the manageability service.

355 A manageability provider advertises, registers, and publishes available manageability endpoints  
356 just like any other Web service endpoints.

357 A manageability consumer establishes which capabilities are supported by the manageable  
358 resource either from the description of the manageability service or by exchanging messages  
359 with the manageability endpoint. For example, a consumer may inspect a WSDL document  
360 looking for manageability operations it is interested in. MUWS and the specifications that build

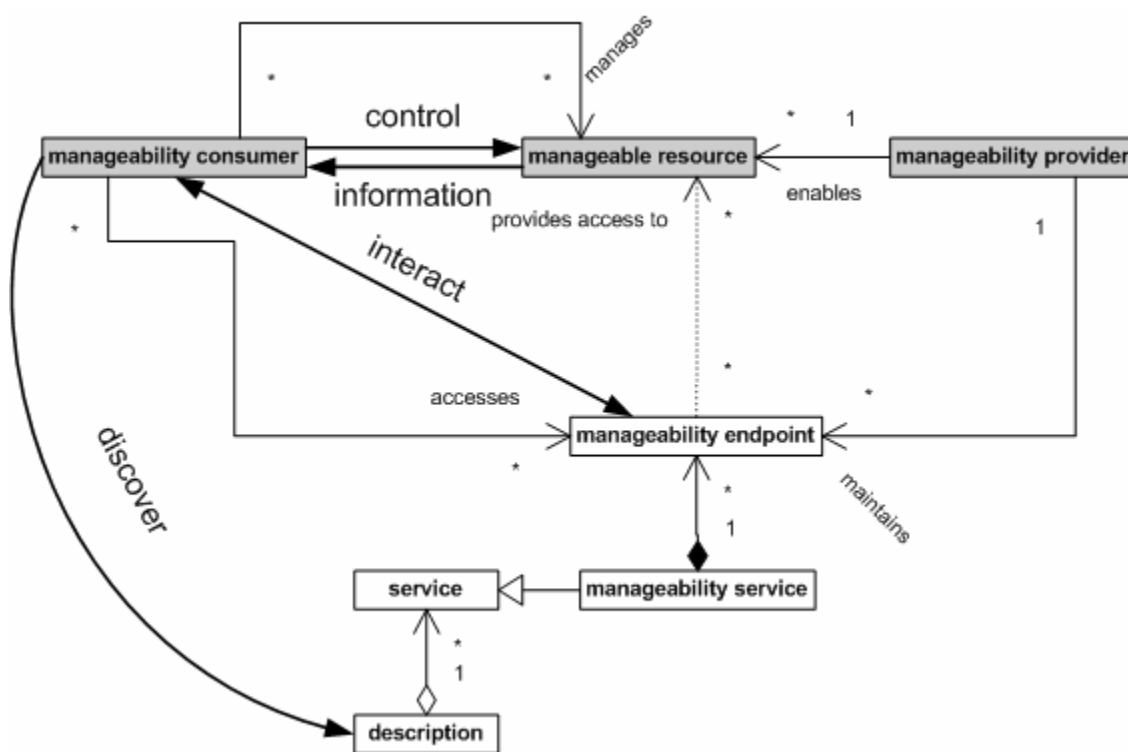
361 on top of it clearly indicate which message Qnames correspond to which operations and which  
 362 capabilities they participate in.

### 363 2.5.3 Interaction

364 A manageability consumer exerts control over, and obtains information about, the manageable  
 365 resource by exchanging messages with one or more manageability endpoints providing access  
 366 to the manageable resource. Message exchanges must match the format and sequence as  
 367 prescribed by the binding description of the endpoint. This interaction is no different than  
 368 exchanging messages with functional Web service endpoints.

369 Figure 2.5.3-1 captures the main principles of the processing model as described above. In this  
 370 context, to interact means to exchange messages.

371



372

373

Figure 2.5.3-1, MUWS Basic Processing Model

374

## 3 Support from Web Services Platform

375 **Management Using Web Services (MUWS)** is a foundation for management of all IT resource  
376 domains. It accomplishes this goal by using Web services features and standards to provide a  
377 flexible, scalable, distributed, and collaborative management framework. An overview of the  
378 features of this framework is provided in Appendix E, Web Services Platform. Appendix E also  
379 provides details about these Web service features, and motivating factors for using Web  
380 services features in MUWS, and provides a non-normative analysis of what standards or  
381 capabilities should be leveraged to support these features.

382 MUWS leverages the Web Services Resources Framework ([WSRF]), in which, the MUWS  
383 manageable resources are represented by Web services as “resources”, in the WSRF sense of  
384 the term. This implies that references to manageability endpoints in MUWS use the mechanism  
385 defined by WSRF, leveraging endpoint references (EPR) as defined by WS-Addressing.

386 If the manageability endpoint corresponds to a variable number (zero or more) of manageable  
387 resources, then the WSRF Implied Resource Pattern **MUST** be followed. This means that the  
388 element(s) listed in the ReferenceProperties of a WS-Resource qualified EPR must be included  
389 in the header of messages sent to such manageability endpoints. This specification does not  
390 currently define how to obtain the EPR. There may be an out-of-band agreement between  
391 provider and consumer on how to obtain EPRs or future versions of this specification would  
392 clarify this subject.

393 In the specific case where a manageability endpoint corresponds to one and only one  
394 manageable resource, then, either the WSRF Implied Resource Pattern, as above, or the  
395 singleton WS-Resource implied pattern **MUST** be used. If the singleton WS-Resource implied  
396 pattern is used, this means that the manageability endpoint does not expect to receive the  
397 elements listed in the ReferenceProperties section of WS-Resource qualified EPRs in the  
398 message headers to indicate which resource is being managed. A manageability consumer who  
399 does not have an EPR for a manageability endpoint **MAY** try to invoke manageability operations  
400 without including reference properties information. If such an invocation succeeds, the  
401 manageability consumer knows it is talking about a manageable resource through a  
402 manageability provider.

403 Further, management properties defined in MUWS are represented as “properties”, in the WSRF  
404 sense of the term, using the mechanisms defined in WS-ResourceProperties ([WS-  
405 ResourceProperties]). This means that each manageable resource exposes a resource  
406 properties document and makes this document available as specified in  
407 WS-ResourceProperties.

408 Supporting WS-ResourceProperties means that any implementation of an interface that includes  
409 properties **MUST** include access methods to these properties as defined by  
410 WS-ResourceProperties. Specifically, the interface **MUST** include the GetResourceProperty  
411 operation defined by WS-ResourceProperties and **MAY** include the GetMultipleProperties,  
412 SetResourceProperties and QueryResourceProperties operations. If the  
413 QueryResourceProperties operation is provided, it **SHOULD** support the XPath 1.0 query  
414 expression dialect, represented by URI <http://www.w3.org/TR/1999/REC-xpath-19991116>.

415 For security, MUWS relies on generic Web services security mechanisms, including  
416 transport-level security and WSS SOAP Message Security as standardized by OASIS. MUWS  
417 1.0 will include a more detailed “security considerations” section.

418 Events are not supported in MUWS 0.5 but will be supported in MUWS 1.0. To do so, MUWS  
419 1.0 will leverage a Web services eventing mechanism.

420

## 4 Manageability Capabilities

421

There is a minimum set of manageability capabilities that the manageability provider must support in order to implement the MUWS specification.

422

423

Manageability capabilities define resource specific properties, operations and events. Details of these manageability capabilities are exposed by the manageable resource.

424

425

A manageable resource MAY also define new resource-specific manageability capabilities.

426

A manageable resource SHOULD extend a MUWS manageability capability when defining a resource-specific manageability capability that uses similar semantics. A manageable resource is not required to extend a MUWS manageability capability when defining a resource-specific manageability capability that uses conflicting semantics.

427

428

429

430

Each capability is formally expressed in a UML diagram using the following approach. Figure 4-1 expresses that a ManageableXSampleCapabilityY is a concept X manageability capability, which is also a manageability capability in a general, conceptual, sense. The name of the capability identifies the distinct semantics that the capability bears. Semantics are then expressed as properties, operations, events and metadata contained in the capability model representation (UML class).

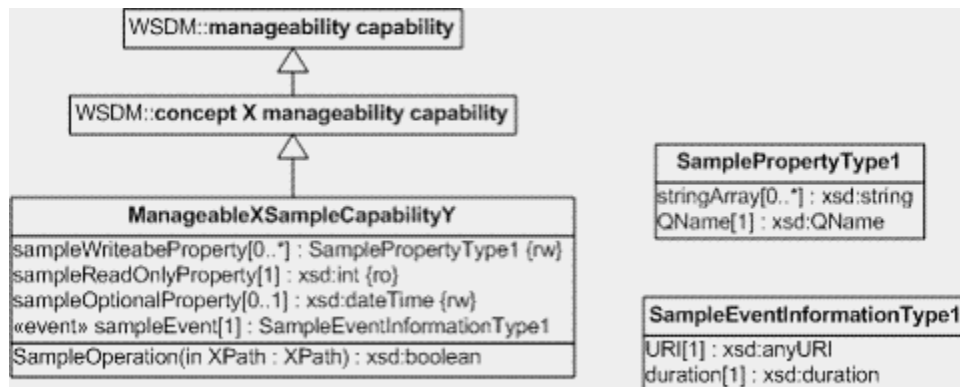
431

432

433

434

435



436

437

Figure 4-1 Manageability Capability UML Sample

438

Properties and operations are expressed as regular UML properties and operations. The meaning of properties and operations is expressed in the text. Properties are defined with, and operations act upon, the information types that are captured by UML classes.

439

440

441

SamplePropertyType1 is an example of such an information type. Simple information types may also be used directly when defining properties and operations. For example, xsd:int is an example of a simple information type that belongs to XML Schema Data Types UML package.

442

443

444

As a simplification, this document uses a convention that all simple information types having a name which name with xsd: belong to that package. XML Schema simple information types, or data types, are defined by the W3C specification at <http://www.w3.org/TR/xmlschema-2/>.

445

446

447

Events are expressed as UML properties with an <<event>> stereotype. The name of the property is the name of the event. The text describes why and when an event occurs and the specific information that is generated or captured when the event occurs. The information type of an event is captured in a UML class which contains proper information element definitions.

448

449

450

451

SampleEventInformationType1 is an example of an event information type.

452 Optionality of properties and events is indicated by multiplicity of the corresponding model  
453 elements: [1] indicates that an element is mandatory, [0..1] indicates that an element is optional.  
454 For array properties, [0..\*] indicates optional and [1..\*] indicates mandatory.

455 The metadata about various model elements is captured as UML constraints. For example,  
456 sampleReadOnlyProperty has an {ro} constraint. This document uses the following common  
457 constraints in the models.

- 458 • ro – means read only, applicable to properties,
- 459 • rw – means read/write, applicable to properties.
- 460 • const – means constant, does not change during runtime, applicable to properties.

461

462 In this section the following namespaces will be used unless specified otherwise. Table 4-1  
463 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
wsdl	http://www.w3.org/2002/07/wsdl
soap	http://schemas.xmlsoap.org/wsdl/soap/
xs	http://www.w3.org/2001/XMLSchema

464 Table 4-1 Manageability Capability Namespaces

465 XML elements and schema types introduced in this section belong to the **muws-xs** namespace.

466 WSDL elements introduced in this section belong to the **muws-wsdl** namespace.

## 467 4.1 Identity

### 468 4.1.1 Definition

469 The goal of Identity is to establish whether two entities are the same. This is a required capability  
470 and it MUST be provided by every manageability service. Observe that this requirement does  
471 not preclude the manageability service using a security policy that prevents some requester from  
472 accessing a capability.

473 In addition, this interface is used as a “marker” interface to allow a consumer to know that the  
474 service that implements it is a manageability service. See section 2.5.2, Discovery, for additional  
475 information.

476 Figure 4.1.1-1 shows the UML representation of MUWS Identity.

477

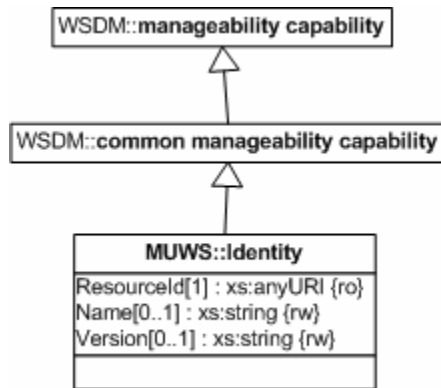


Figure 4.1.1-1 MUWS Identity

478

479

## 480 4.1.2 Data Types

481 This specification does not define any data type to represent Identity.

## 482 4.1.3 Properties

483 Following is the specification of the resource Identity properties (elements).

484

```

485 <ResourceId>xsd:anyURI</ResourceId>
486 <Name>xsd:string</Name>?
487 <Version>xsd:string</Version>?
  
```

488

489 Following is an example excerpt from a resource properties instance document that contains  
490 these properties.

```

491 <ResourceId>http://example.com/resource/diskDrive/9F34AD35B</ResourceI
492 d>
493 <Name>The disk drive in Bob's laptop</Name>
494 <Version>1.0</Version>
  
```

495

496 **ResourceId** is an opaque identifier of the resource managed through the manageability  
497 endpoint. It is a read-only mandatory property that has a cardinality of 1.

498 The following constraints are applicable to ResourceId:

- 499 • Globally unique: A manageability endpoint **MUST** create the ResourceId URI in a way  
500 that ensures that the ResourceId is unique to the resource managed through the  
501 manageability endpoint and globally unique.. This specification does not prescribe the  
502 means by which global uniqueness is achieved.
- 503 • Uniqueness in time: A ResourceId **MUST NOT** be reused by any manageability endpoint  
504 for another resource, even after the original resource no longer exists.
- 505 • Consistency across endpoints: A manageability provider **SHOULD** use the ResourceId  
506 that is suggested by the characteristics of the resource to identify the resource. This is  
507 for example possible when the ResourceId can be retrieved from the resource by the  
508 manageability endpoint or when an application of MUWS to a given domain specifies a

509 method to build the ResourceId based on characteristics of the resources within the  
510 domain. It is not guaranteed that different manageability endpoints attached to the same  
511 resource will, in all cases, return the same ResourceId

512 • Consistency within an endpoint: A manageability provider that exposes several  
513 manageability endpoints for the same resource SHOULD use the same ResourceId for  
514 all manageability endpoints.

515 • Persistence: A manageability endpoint SHOULD return the same ResourceId during the  
516 entire lifetime of the manageability endpoint, including across power cycles of the  
517 manageability endpoint. Resources which are not able to persist the ResourceId across  
518 power cycles of the manageability endpoint SHOULD try to provide a consistent  
519 ResourceId via predictable identifier generation or delegation of Identity assignment.  
520 Manageability consumers may not be able to determine if two manageable resources  
521 which do not provide the same ResourceId correspond to the same resource and as a  
522 result a single resource may be treated as many resources.

523 • Equality: If the ResourceIds are equal then the consumer MUST assume that the two  
524 manageability endpoints represent the same resource. However, a manageability  
525 consumer MUST NOT assume that two manageability endpoints are representing two  
526 different resources solely because the ResourceId is different. Two different ID's could  
527 conceivably reference the same resource. It is strongly recommended that this condition  
528 be avoided in a conscious and deliberate manner, as some managers may not be able to  
529 distinguish that the two identifiers are, in fact, attached to the same resource. Thus, the  
530 managers would be forced to treat every identifier as an attachment to an unique  
531 resource.

532 Since the ResourceId is defined as opaque, this specification does not allow the consumer to  
533 infer any characteristic of the resource by examining the ResourceId, other than comparing  
534 the ResourceId to another ResourceId as one way to establish oneness. For example, one  
535 possible way to construct the ResourceId and ensure its uniqueness is to use a UUID  
536 wrapped in a URI.

537 Note that this specification does not define equivalence of URIs and the consumer should decide  
538 which level of the comparison ladder defined in section 6 of [RFC2396bis] is appropriate to use  
539 for this comparison.

540 The following two paragraphs describe a mechanism, intended to be introduced in MUWS 1.0,  
541 called "correlatable names". This mechanism is not available in MUWS 0.5.

542 The correlatable names mechanism is one of several possible mechanisms that can be used  
543 when the ResourceId mechanism cannot provide certainty about the oneness of two resources.  
544 The correlate management capability can be used by a resource manager to determine if two  
545 different resourceIds, produced at different times, by the same manageability provider for an  
546 endpoint, represents the same endpoint.

547 The basic idea is to provide a list of properties that, if all are equal between two manageability  
548 endpoints, guarantees that the resource(s) behind the manageability endpoints are one. In the  
549 case where two ResourceIds match but the correlatable names do not, a case that is not allowed,  
550 but can not be guaranteed never to happen, the resource is considered to be the same. This is  
551 because the ResourceId has precedence over the correlatable name.

552 **Name** is a string containing a descriptive name for the resource being managed. The *Name* is  
553 intended for human consumption. It is a read-write optional property that has a cardinality of 0..1.

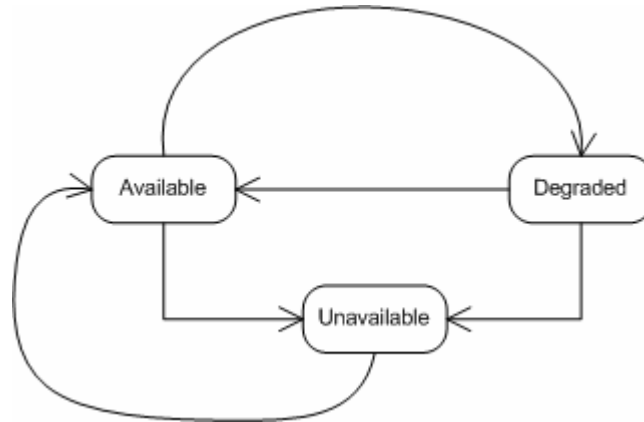
554 **Version** is a string representing the version of the resource being managed. MUWS does not  
555 specify how this string is constructed. The version string can be specified by any domain-specific  
556 specification that use MUWS. Version is a read-write optional property with a cardinality of 0..1.

557 **4.2 State**

558 **4.2.1 Definition**

559 The goal of this section is to define a state model for any IT resource and state management  
560 capabilities through Web services. The state model must support the state change capabilities  
561 and the events for lifecycle and status changes. Additionally, the state model must be extensible.

562 Figure 4.2.1-1 shows the resource state model without any sub-states.

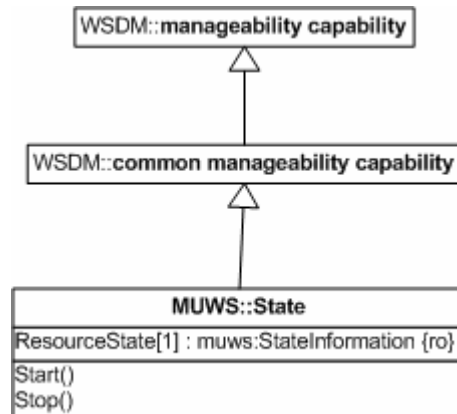


563

564

Figure 4.2.1-1 Resource State Model

565 Figure 4.2.1-2 shows the UML representation of MUWS State.



566

567

Figure 4.2.1-2 MUWS State

568 **4.2.2 Description of State Model**

569 Universal Resource Identifiers (URI) for the valid top-level resource states :

570

- <http://docs.oasis-open.org/wsdm/2004/04/muws/state/available>

571

This URI corresponds to the **Available** state in the UML diagram. A resource in the **Available** state is able to perform all of its functional tasks.

572

573

- <http://docs.oasis-open.org/wsdm/2004/04/muws/state/degraded>

574

This URI corresponds to the **Degraded** state in the UML diagram. A resource in this **Degraded** state is able to perform some, but not all, of its functional tasks.

575

- 576 • <http://docs.oasis-open.org/wsdm/2004/04/muws/state/unavailable>  
 577 This URI corresponds to the **Unavailable** state in the UML diagram. A resource in this  
 578 state is not able to perform any of its functional tasks.  
 579

580 The resource state model, as defined in this specification, identifies three generic states that are  
 581 relevant for most, if not all, types of resources. Implementers SHOULD reuse a defined state if it  
 582 is appropriate for their resource. It is anticipated that new states, and corresponding URIs, may  
 583 need to be defined. For example, the need for a new state may be indicated when the existing  
 584 states do not represent the intended meaning for a resource, or, when the existing states are too  
 585 general and the implementer wishes to expose a more specific and useful state description. Just  
 586 as implementers MAY create new states, implementers MAY create new transitions between any  
 587 two states. Note, the two states involved in the transition need not be defined within the same  
 588 organization. Also, note that until MUWS defines a way to represent the state model for a  
 589 resource, "creating a new transition" just means writing some text to explain that a resource state  
 590 can change from one state to another. Finally, please note that we expected MUWS 1.0 will  
 591 define a mechanism enabling implementers to define resource states as sub-states of another  
 592 state, and, will specify the recommendations for its use.

593 The following table lists the valid transitions between the top-level resource states.

594

Start State	End State
Available	Degraded
Available	Unavailable
Degraded	Available
Degraded	Unavailable
Unavailable	Available

595

### 596 4.2.3 Data Types

597 The following is a pseudo-schema fragment declaring reusable data types for managing the  
 598 resource state.

599

```
600 <xs:complexType name="StateInformation">
601   <xs:sequence>
602     <xs:element name="State" type="xs:anyURI"/>
603     <xs:element name="TimeEntered" type="xs:dateTime"/>
604   </xs:sequence>
605 </xs:complexType>
```

606

607 The **(StateInformation)** type contains information about a resource state.

608 The **(StateInformation)/State** element provides the URI of a resource state.

609 The **(StateInformation)/TimeEntered** element provides the time the resource entered the  
 610 identified state.

## 611 4.2.4 Properties

612 The resource state properties, or elements, are specified as follows:

613

```
614 <ResourceState>StateInformation</ResourceState>
```

615

616 The following fragment provides an example from a resource properties instance document  
617 containing this property:

```
618 <ResourceState>  
619   <State>  
620     http://docs.oasis-open.org/wsdm/2004/04/muws/state/available  
621   </State>  
622   <TimeEntered>2004-03-11T11:30:56Z</TimeEntered>  
623 </ResourceState>
```

624

625 The **ResourceState** property contains the current state of the resource. This property is a read-  
626 only and mandatory with a cardinality of 1

## 627 4.2.5 Operations

628 This section describes the messages for performing operations on the state of a resource.

### 629 4.2.5.1 Start

630 Request:

```
631 <Start/>
```

632

633 Reply:

```
634 <StartOK/>
```

635

636 Upon receiving a request for a Start operation, a manageability provider attempts to transition  
637 the resource state from Unavailable to Available, according to the state model. If the transition  
638 completes, then the operation also completes successfully. If the initial resource state is  
639 Available, this operation completes successfully.

### 640 4.2.5.2 Stop

641 Request:

```
642 <Stop/>
```

643

644 Reply:

```
645 <StopOK/>
```

646

647 Upon receiving a request for a Stop operation, a manageability provider attempts to change the  
648 resource state from Available, or Degraded, to Unavailable according to the state model. If the  
649 transition completes, this operation completes successfully. If the initial resource state is  
650 Unavailable, this operation completes successfully.

651

## 652 4.3 Metrics

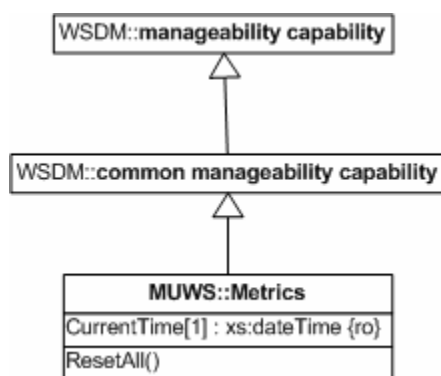
### 653 4.3.1 Definition

654 Metrics are a specific type of property. Metrics represent collected values and have a related  
655 collection period.

656 The goal of this section is to describe the characteristics of a typical property used to represent  
657 collected numerical data, called Metrics. A common characteristic of metrics is that they change  
658 over time and can be reset.

659 Figure 4.3.1-1 presents the Metrics capability in context.

660



661

662

Figure 4.3.1-1 MUWS Metrics

663 As a simple example, consider a toll bridge with two properties; the length of the bridge and the  
664 number of cars that have passed over the bridge. The length of the bridge, while numeric, is not  
665 a metric; it represents the current configuration of the bridge. You can not reset the length of the  
666 bridge. By contrast, the number of cars that have passed over the bridge is a metric. It requires  
667 collecting, or counting, the number of cars typically for some duration of time, such as the last  
668 hour, the last day or even since the bridge was constructed. You can reset the number of cars,  
669 for example, at the start of a new interval.

670 When accessing metrics, the time associated with the metric is typically required. In the  
671 example, above it would be useful to know that the number “100” represents the number of cars  
672 that have passed over the bridge in the last fifteen minutes. Similarly, if data is posted on a  
673 periodic basis, it may be useful to know the time of last update.. For this reason, the standard  
674 data type for Metrics has reset-time and update-time attributes. However, both are optional.

675 When looking at a value, it is important to have a notion of how changes to that metric are  
676 interpreted. That notion is defined as a *change type*. Metrics have two (2) change types:

- 677
- Counter, increments with usage
- 678
- Gauge, moves between a range of values

679 Metrics can be reset either periodically, such as 'once a day', or on demand. The meaning of  
680 resetting a metric varies with each metric. The meaning of a metric SHOULD be clarified as  
681 needed, in the description of a metric. Often, resetting a metric means setting its base or initial  
682 value, typically zero, but this is not mandatory. (Note that this specification provides no specific  
683 means for the scheduling of reset operations.)

## 684 4.3.2 Data Types

685 The following schema fragment declares the (reusable) data types used to expose the metrics of  
686 the resource. All attributes defined in the **MetricAttributes** attribute group are optional.

687

```
688 <xs:attributeGroup name="MetricAttributes">
689   <xs:attribute name="ResetAt" type="xs:dateTime" />
690   <xs:attribute name="LastUpdated" type="xs:dateTime" />
691   <xs:attribute name="ChangeType">
692     <xs:simpleType>
693       <xs:restriction base="xs:string">
694         <xs:enumeration value="Counter" />
695         <xs:enumeration value="Gauge" />
696       </xs:restriction>
697     </xs:simpleType>
698   </xs:attribute>
699   <xs:attribute name="TimeScope">
700     <xs:simpleType>
701       <xs:restriction base="xs:string">
702         <xs:enumeration value="Interval" />
703         <xs:enumeration value="PointInTime" />
704         <xs:enumeration value="SinceReset" />
705       </xs:restriction>
706     </xs:simpleType>
707   </xs:attribute>
708   <xs:anyAttribute namespace="##other" processContents="lax" />
709 </xs:attributeGroup>
```

710

711 **(MetricAttributes)** attribute group MUST be included in every metric type or metric property  
712 element declaration.

713 **(MetricAttributes)/ResetAt** indicates the time when a particular metric was reset. UTC or Z-  
714 coded can be reported.

715 **(MetricAttributes)/LastUpdated** indicates the last update time of a metric value.

716 **(MetricAttributes)/ChangeType** indicates a type of change pattern supported by a metric.

717 § **Counter** declares a metric value. that can only increase

718 § **Gauge** declares a metric value that can increase or decrease

719 **(MetricAttributes)/TimeScope** indicates the time interval used to calculate a metric.

720 § **Interval** declares a metric value that is calculated within a certain time interval. Usually  
721 the interval is defined by the specification of the metric property. For example,  
722 RequestsPerSecond is an interval metric.

723 § **PointInTime** declares a metric value that is calculated when the metric property is  
724 retrieved. For example, CurrentTemperature is a point-in-time metric.

725 § **SinceReset** declares a metric value that is calculated since the **ResetAt** time mark.

726 The following three types are defined for metrics that are integers, durations, or times.

727 Specifications that use MUWS to create manageability properties applicable to specific domains

728 are encouraged to use these types, or to create new metric types, by including the  
729 **MetricAttributes** attribute group in created metric types.

```
730 <xs:complexType name="IntegerMetric">  
731   <xs:simpleContent>  
732     <xs:extension base="xs:integer">  
733       <xs:attributeGroup ref="muws-xs:MetricAttributes"/>  
734       <xs:anyAttribute namespace="##other" />  
735     </xs:extension>  
736   </xs:simpleContent>  
737 </xs:complexType>
```

739

740 **(IntegerMetric)** type declares an xsd:integer metric.

741

```
742 <xs:complexType name="DurationMetric">  
743   <xs:simpleContent>  
744     <xs:extension base="xs:duration">  
745       <xs:attributeGroup ref="muws-xs:MetricAttributes"/>  
746       <xs:anyAttribute namespace="##other" />  
747     </xs:extension>  
748   </xs:simpleContent>  
749 </xs:complexType>
```

751

752

753 **(DurationMetric)** type declares an xs:duration metric.

### 754 4.3.3 Properties

755 The following provides the specification of a resource metrics property.

756

```
757 <CurrentTime>xs:dateTime</CurrentTime>
```

758

759 The following is an example fragment of a resource properties instance document containing this  
760 property.

761

```
762 <CurrentTime>2004-03-11T11:30:56Z</CurrentTime>
```

763

764 **CurrentTime** contains the time it was retrieved from the manageability provider. This property is  
765 useful to manageability consumers when analysing time values received from a manageability  
766 endpoint in the absence of a time synchronization mechanism. It is a read-only mandatory  
767 property that has a resource cardinality of 1.

768 The Metrics capability requires the *CurrentTime* property is present in the resource properties,  
769 and provides a reference point for time-based attributes as defined by metric data types. Note  
770 that *CurrentTime* is not a metric, Rather, it is a property of type xsd:dateTime defined as part of  
771 the "Metrics" capability, consequently, *ResetAll()* operations have no effect on *CurrentTime*.)

772 **4.3.4 Operations**

773 The following messages can be exchanged to perform operations on resource metrics.

774 **4.3.4.1 ResetAll**

775 Request:

776 <**ResetAll**/>

777

778 Reply:

779 <**ResetAllOK**/>

780

781 Upon receiving a *ResetAll* request, a manageability provider resets the value for each of its  
782 metrics. . A subset of the metrics for a resource can be organized as a logical group and reset by  
783 other mechanisms. However, a *ResetAll* request indicates that all metrics for a resource  
784 SHOULD be reset. Note that a manageability consumer MUST NOT assume that different  
785 metrics are reset at the same time, even if they are provided by the same manageability  
786 endpoint.

787

---

## 5 Discovery and Introspection

788 Many forms of discovery are supported by Web services. This specification does not prescribe a  
789 normative method for discovering manageability services. It is expected that discovery methods  
790 as commonly used for Web services will be used as discovery methods for manageability  
791 services.

792 There exists but one normative requirement relative to discovering manageability services, as  
793 follows: A manageability service **MUST** provide the Identity capability through the corresponding  
794 WSDL interface as defined by MUWS. As a result of this requirement, a consumer can inspect  
795 the WSDL description for a Web service and determine if the discovered service acts as a  
796 manageability service.. If the discovered service implements the Identity interface defined by  
797 MUWS, then it is a manageability service..

798

---

## 6 Defining a Manageability Interface

799

The following are normative statements for MUWS representation.

800

- WSDL 1.1 must be used.

801

802

- Additionally, WSDM defines portTypes which have to be combined into a custom portType, operation by operation, according to the WSDL specification.

803

- Document/literal binding must be used.

804

805

- MUWS defines property elements which have to be combined into a custom properties document according to WS-ResourceProperties specification

806

---

807 **7 References**

808 **7.1 Normative**

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

811

812 **[XML Schema Part 1]**

813 Henry S. Thompson, et al. *XML Schema Part 1: Structures*, W3C  
814 Recommendation, May 2001, <http://www.w3.org/TR/xmlschema-1/>

815

816 **[XML Schema Part 2]**

817 Paul V. Biron, et al. *XML Schema Part 2: Datatypes*, W3C  
818 Recommendation, May 2001, <http://www.w3.org/TR/xmlschema-2/>

819

820 **[XML1.0 3<sup>rd</sup> Edition]**

821 Tim Bray, et al., *Extensible Markup Language (XML) 1.0 (Third Edition)*,  
822 W3C Recommendation, February 2004, <http://www.w3.org/TR/REC-xml>

823

824 **[XNS]**

825 Tim Bray, et al., *Extensible Namespaces in XML*, W3C  
826 Recommendation, January 1999, [http://www.w3.org/TR/REC-xml-  
names/](http://www.w3.org/TR/REC-xml-names/)

827

828 **[SOAP1.1]**

829 Don Box, et al., *Simple Object Access Protocol (SOAP) 1.1*, W3CNote,  
830 May 2000, <http://www.w3.org/TR/soap11/>

831

832 **[WSDL1.1]**

833 Erik Christensen, et al., *Web services Description Language (WSDL)*  
834 *1.1*, W3C Note, March 2001, <http://www.w3.org/TR/wsdl>

835

836 **[UDDI]**

837 Tom Bellwood, et al., *Web UDDI Version 3.0*, UDDI Spec Technical  
838 Committee Specification, July 2003, [http://uddi.org/pubs/uddi-v3.00-  
published-20020719.htm](http://uddi.org/pubs/uddi-v3.00-published-20020719.htm)

839

840 **[WSA]**

841 David Booth, et al. *Web Services Architecture*, W3C Working Group  
842 Note, February 2004, [http://www.w3.org/TR/2004/NOTE-ws-arch-  
20040211/](http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/)

843

844 **[WSRF]**

845 Karl Czajkowski, et al. *The WS-Resource Framework version 1.0*,  
846 [http://devresource.hp.com/drc/specifications/wsrf/WSRF\\_overview-1-  
0.pdf](http://devresource.hp.com/drc/specifications/wsrf/WSRF_overview-1-0.pdf) and related documents available at  
847 <http://devresource.hp.com/drc/specifications/wsrf/index.jsp>

848

849 **[WS-ResourceProperties]**

848 Steve Graham, et al., *Web service Resource Properties version 1.1*,  
849 January 2004, [http://devresource.hp.com/drc/specifications/wsrp/WS-](http://devresource.hp.com/drc/specifications/wsrp/WS-ResourceProperties-1-1.pdf)  
850 [ResourceProperties-1-1.pdf](http://devresource.hp.com/drc/specifications/wsrp/WS-ResourceProperties-1-1.pdf)

851

852 **[WS-Addressing]** Don Box, et al., *Web services Addressing*, March 2003,  
853 [http://msdn.microsoft.com/library/default.asp?url=/library/en-](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-addressing.asp)  
854 [us/dnglobspec/html/ws-addressing.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-addressing.asp)

855

856

## 857 **7.2 Non-normative**

858 **[RFC2396bis]** T. Berners-Lee, et al., *Uniform Resource Identifier (URI): Generic*  
859 *Syntax*, IETF RFC 2396bis-04, February 2004,  
860 <http://www.ietf.org/internet-drafts/draft-fielding-uri-rfc2396bis-04.txt>

861

862 **[MUWS REQS]** Pankaj Kumar, et al., *Requirements – Management Using Web*  
863 *Services*, Committee Draft, October 2003, [http://www.oasis-](http://www.oasis-open.org/apps/org/workgroup/wsdm/download.php/6185/WSDM-MUWS-Req-committee-draft-1.0-20031002.pdf)  
864 [open.org/apps/org/workgroup/wsdm/download.php/6185/WSDM-MUWS-](http://www.oasis-open.org/apps/org/workgroup/wsdm/download.php/6185/WSDM-MUWS-Req-committee-draft-1.0-20031002.pdf)  
865 [Req-committee-draft-1.0-20031002.pdf](http://www.oasis-open.org/apps/org/workgroup/wsdm/download.php/6185/WSDM-MUWS-Req-committee-draft-1.0-20031002.pdf)

866

---

## 867 Appendix A. Acknowledgements

868 The following people made contributions to this specification:

- 869 • Winston Bumpus <winston\_bumpus@dell.com>
- 870 • Brian Carol <brian.carroll@merant.com>
- 871 • Fred Carter <fred.carter@amberpoint.com>
- 872 • John DeCarlo <jdecarlo@mitre.org>
- 873 • Andreas Dharmawan <andreas@westbridgetech.com>
- 874 • Mark Ellison <ellison@ieee.org>
- 875 • Heather Kreger <kreger@us.ibm.com>
- 876 • Hal Lockhart <hlockhar@bea.com>
- 877 • Bryan Murray <bryan.murray@hp.com>
- 878 • Richard Nikula <Richard\_Nikula@bmc.com>
- 879 • Micheal Perks <mperks@us.ibm.com>
- 880 • Homayoun Pourheidari <homayoun@hp.com>
- 881 • Karl Schopmeyer <k.schopmeyer@attglobal.net>
- 882 • Igor Sedukhin <Igor.Sedukhin@ca.com>
- 883 • Ellen Stokes <stokese@us.ibm.com>
- 884 • William Vambenepe <william\_vambenepe@hp.com>
- 885 • Andrea Westerinen <andreaw@cisco.com>

886

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

- 889 • Guru Bhat <Guru.Bhat@oracle.com>
- 890 • Jeff Bohren <jbohren@opennetwork.com>
- 891 • Winston Bumpus <winston\_bumpus@dell.com>
- 892 • Fred Carter <fred.carter@amberpoint.com>
- 893 • John DeCarlo <jdecarlo@mitre.org>
- 894 • Andreas Dharmawan <andreas@westbridgetech.com>
- 895 • Mark Ellison <ellison@ieee.org>
- 896 • Daniel Foody <dan@actional.com>
- 897 • Heather Kreger <kreger@us.ibm.com>

- 898 • Bryan Murray <bryan.murray@hp.com>
- 899 • Paul Lipton <paul.lipton@ca.com>
- 900 • Hal Lockhart <hlockhar@bea.com>
- 901 • Rajiv Maheshwari <rajiv.k.maheshwari@oracle.com>
- 902 • Richard Nikula <Richard\_Nikula@bmc.com>
- 903 • Michael Perks <mperks@us.ibm.com>
- 904 • Homayoun Pourheidari <homayoun@hp.com>
- 905 • Karl Schopmeyer <k.schopmeyer@attglobal.net>
- 906 • Igor Sedukhin <Igor.Sedukhin@ca.com>
- 907 • Davanum Srinivas <Davanum.Srinivas@ca.com>
- 908 • Ellen Stokes <stokese@us.ibm.com>
- 909 • Thomas Studwell <studwell@us.ibm.com>
- 910 • Ryoichi Ueda <ueda@sdl.hitachi.co.jp>
- 911 • William Vambenepe <william\_vambenepe@hp.com>
- 912 • Andrea Westerinen <andreaw@cisco.com>

913

914 Concepts for the "Metrics" section were inspired by work from the DMTF applications/Metrics  
915 WG.

916

---

## Appendix B. Notices

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

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

929 Copyright © OASIS Open 2003. *All Rights Reserved.*

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

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

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

946

947

948

949

950

---

## Appendix C. Schemas (Normative)

951

```
952 <?xml version="1.0" encoding="utf-8"?>
953 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
954           xmlns:muws-xs="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/schema"
955           targetNamespace="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/schema"
956           elementFormDefault="qualified" attributeFormDefault="unqualified">
957
958   <xs:element name="ResourceId" type="xs:anyURI"/>
959   <xs:element name="Name" type="xs:string"/>
960   <xs:element name="Version" type="xs:string"/>
961
962   <xs:complexType name="StateInformation">
963     <xs:sequence>
964       <xs:element name="State" type="xs:anyURI"/>
965       <xs:element name="TimeEntered" type="xs:dateTime"/>
966     </xs:sequence>
967   </xs:complexType>
968
969   <xs:element name="ResourceState" type="muws-xs:StateInformation"/>
970
971   <xs:attributeGroup name="MetricAttributes">
972     <xs:attribute name="ResetAt" type="xs:dateTime"/>
973     <xs:attribute name="LastUpdated" type="xs:dateTime"/>
974     <xs:attribute name="ChangeType">
975       <xs:simpleType>
976         <xs:restriction base="xs:string">
977           <xs:enumeration value="Counter"/>
978           <xs:enumeration value="Gauge"/>
979         </xs:restriction>
980       </xs:simpleType>
981     </xs:attribute>
982     <xs:attribute name="TimeScope">
983       <xs:simpleType>
984         <xs:restriction base="xs:string">
985           <xs:enumeration value="Interval"/>
986           <xs:enumeration value="PointInTime"/>
987           <xs:enumeration value="StartupInterval"/>
988         </xs:restriction>
989       </xs:simpleType>
990     </xs:attribute>
991     <xs:anyAttribute namespace="##other" processContents="lax"/>
992   </xs:attributeGroup>
993
994   <xs:complexType name="IntegerMetric">
995     <xs:simpleContent>
996       <xs:extension base="xs:integer">
997         <xs:attributeGroup ref="muws-xs:MetricAttributes"/>
998         <xs:anyAttribute namespace="##other" processContents="lax"/>
999       </xs:extension>
1000     </xs:simpleContent>
1001   </xs:complexType>
```

```

1002
1003 <xs:complexType name="DurationMetric">
1004     <xs:simpleContent>
1005         <xs:extension base="xs:duration">
1006             <xs:attributeGroup ref="muws-xs:MetricAttributes"/>
1007             <xs:anyAttribute namespace="##other" processContents="lax"/>
1008         </xs:extension>
1009     </xs:simpleContent>
1010 </xs:complexType>
1011
1012 <xs:element name="CurrentTime" type="xs:dateTime"/>
1013
1014 <xs:complexType name="ResourceIdentityPropertiesType">
1015     <xs:sequence>
1016         <xs:element ref="muws-xs:ResourceId"/>
1017         <xs:element ref="muws-xs:Name" minOccurs="0"/>
1018         <xs:element ref="muws-xs:Version" minOccurs="0"/>
1019         <xs:any minOccurs="0" maxOccurs="unbounded"
1020             namespace="##other" processContents="lax"/>
1021     </xs:sequence>
1022 </xs:complexType>
1023
1024 <xs:element name="ResourceIdentityProperties"
1025     type="muws-xs:ResourceIdentityPropertiesType"/>
1026
1027 <xs:complexType name="ResourceStatePropertiesType">
1028     <xs:sequence>
1029         <xs:element ref="muws-xs:ResourceState"/>
1030         <xs:any minOccurs="0" maxOccurs="unbounded"
1031             namespace="##other" processContents="lax"/>
1032     </xs:sequence>
1033 </xs:complexType>
1034
1035 <xs:element name="ResourceStateProperties"
1036     type="muws-xs:ResourceStatePropertiesType"/>
1037
1038 <xs:complexType name="ResourceMetricsPropertiesType">
1039     <xs:sequence>
1040         <xs:element ref="muws-xs:CurrentTime"/>
1041         <xs:any minOccurs="0" maxOccurs="unbounded"
1042             namespace="##other" processContents="lax"/>
1043     </xs:sequence>
1044 </xs:complexType>
1045
1046 <xs:element name="ResourceMetricsProperties"
1047     type="muws-xs:ResourceMetricsPropertiesType"/>
1048
1049 <xs:element name="Start"><xs:complexType/></xs:element>
1050 <xs:element name="StartOK"><xs:complexType/></xs:element>
1051
1052 <xs:element name="Stop"><xs:complexType/></xs:element>
1053 <xs:element name="StopOK"><xs:complexType/></xs:element>
1054

```

```
1055 <xs:element name="ResetAll"><xs:complexType/></xs:element>
1056 <xs:element name="ResetAllOK"><xs:complexType/></xs:element>
1057
1058 </xs:schema>
```

---

## Appendix D. WSDL elements (Normative)

1059

```
1060 <?xml version="1.0" encoding="utf-8"?>
1061 <definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
1062     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
1063     xmlns:xs="http://www.w3.org/2001/XMLSchema"
1064     xmlns:wsrp="http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
1065     xmlns:muws-xs="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/schema"
1066     xmlns:muws-wsdl="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/wsdl"
1067     targetNamespace="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/wsdl">
1068
1069     <types>
1070     <xs:schema elementFormDefault="qualified"
1071         targetNamespace="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/wsdl">
1072
1073         <xs:import namespace="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/schema"
1074             schemaLocation="http://docs.oasis-open.org/wsdm/2004/04/muws-0.5/schema"/>
1075
1076     </xs:schema>
1077     </types>
1078
1079     <message name="StartRequest">
1080         <part name="body" element="muws-xs:Start"/>
1081     </message>
1082
1083     <message name="StartResponse">
1084         <part name="body" element="muws-xs:StartOK"/>
1085     </message>
1086
1087     <message name="StopRequest">
1088         <part name="body" element="muws-xs:Stop"/>
1089     </message>
1090
1091     <message name="StopResponse">
1092         <part name="body" element="muws-xs:StopOK"/>
1093     </message>
1094
1095     <message name="ResetAllRequest">
1096         <part name="body" element="muws-xs:ResetAll"/>
1097     </message>
1098
1099     <message name="ResetAllResponse">
1100         <part name="body" element="muws-xs:ResetAllOK"/>
1101     </message>
1102
1103     <portType name="Identity"
1104         wsrp:ResourceProperties="muws-xs:IdentityProperties"/>
1105
1106     <portType name="ResourceState"
1107         wsrp:ResourceProperties="muws-xs:ResourceStateProperties">
1108         <operation name="Start">
1109             <input name="StartRequest" message="muws-wsdl:StartRequest"/>
```

```
1110     <output name="StartResponse" message="muws-wsdl:StartResponse"/>
1111 </operation>
1112 <operation name="Stop">
1113     <input name="StopRequest" message="muws-wsdl:StopRequest"/>
1114     <output name="StopResponse" message="muws-wsdl:StopResponse"/>
1115 </operation>
1116 </portType>
1117
1118 <portType name="Metrics"
1119     wsrp:ResourceProperties="muws-xs:MetricsProperties">
1120     <operation name="ResetAll">
1121         <input name="ResetAllRequest" message="muws-wsdl:ResetAllRequest"/>
1122         <output name="ResetAllResponse" message="muws-wsdl:ResetAllResponse"/>
1123     </operation>
1124 </portType>
1125 </definitions>
1126
```

1127

---

## Appendix E. Web Services Platform

1128

This section briefly describes the Web services platform features that are required in order to specify Management Using Web Services (MUWS), and makes recommendations on the support of each feature. Recommendations made in this section will either:

1129

1130

1131

- Reference another specification to be used to provide the feature

1132

- Identify the feature as something that needs to be provided by the industry in the future

1133

- Identify the feature as something that needs to be defined in a WSDM TC interim specification until such a feature is available for the Web services platform.

1134

1135

### Initial Focus

1136

The Web services platform used by this version of MUWS must support the features listed below.

1137

1138

### Properties

1139

Properties are describable information that can be queried and may also be set.. Properties, and their associated messages and operations, may be provided by a Web service interface in a schema document. A Property consist of a declaration, or name, and a description, or type.

1140

1141

Properties should be introspectable at design-time and at run-time. By using a property schema, it is possible to find, to read, and to write a property.

1142

1143

1144

For manageability, a property is part of the advertised manageability interface for a resource. For manageability, a property can represent configuration values, metrics, identifiers, and so on.

1145

1146

**Motivation:** Many manageability capabilities of a manageable service concern the resource the manager is able to query and set. This information should be modeled as a set of properties, and their access methods. Note that there is a need for access methods that meet the scalability requirements of management applications, for example "bulk-get".

1147

1148

1149

1150

**Recommendation:** Use the WS-Resource Properties specification [WS-ResouceProperty] to describe the properties of a manageable resource.

1151

1152

### Meta Data

1153

Meta data is generally defined as data about data. In the context of management, it is additional descriptive information about the components of a manageability interface. Meta data may apply to the properties, operations, events, and capabilities of the manageability interface, or, meta data may apply to the context, quality, condition, and character of referenced data. Meta data can be introspected at design-time and at run-time.

1154

1155

1156

1157

1158

**Motivation:** In IT management, meta data is important for:

1159

- describing data in richer ways, and ultimately, to link data with the goals driving the existence of the source of this data, For example, meta data concerning the limitations, purpose, context, quality, and character of management data helps describe how the associated data relates to the objectives of an IT environment

1160

1161

1162

1163

- enabling the interoperability of manageability capabilities, as provided by numerous, different providers

1164

1165 **Recommendation:** Descriptive information about the manageability interface will be expressed  
1166 as XML element attributes as a tactical solution for WSDM 0.5. This satisfies run-time  
1167 introspection, but does not provide for design-time introspection.

1168 A strategic solution will be developed for WSDM 1.0 supporting run-time and design-time  
1169 introspection. The concept of a qualifier, as described in the DMTF Common Information Model  
1170 (CIM), and consisting of meta-data concerning a class, property, method, notification or method  
1171 parameter may be mined for useful ideas for manageability meta-data..

## 1172 **Addressing**

1173 An address, or reference, is a data structure referencing a unique Web service. Addressing may  
1174 be used to reference the Web services of a manageability provider, or, to reference a  
1175 manageable resource. This data structure must hold sufficient information for a manageability  
1176 consumer to to locate the Web service and to exchange messages with the Web service. When  
1177 the referenced Web service provides manageability for several resources, the reference data  
1178 structure needs to uniquely identify a specific resource. The reference data structure must  
1179 include data needed to locate the description of its Web service. This enables the manageability  
1180 consumer to identify the messages understood by the Web service. In this context, address and  
1181 reference are used synonymously.

1182 **Motivation:** Manageability consumers need interoperable addressing to reference common  
1183 manageability services and manageable resources, as a reference for relationships, in  
1184 notifications and messages.

1185 **Recommendation:** Use the WS-Addressing specification  
1186 ([http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-addressing.asp)  
1187 [addressing.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-addressing.asp)).

## 1188 **Notification**

1189 Notification is a method of conveying information from a source to recipients expressing interest.  
1190 In terms of Web services, a notification means delivering an XML message from the source to  
1191 addressable recipients. An interest in receiving notifications must be established with the source  
1192 by the recipient, or by a third party on behalf of the recipient. When registering interest, the  
1193 address(es) of the recipient(s) must be provided (see Addressing, above).

1194 **Motivation:** Manageable resources need to convey information to the managers. In certain  
1195 cases, it is unreasonable for the manager to explicitly poll (request) the information, and it has to  
1196 be sent to the manager by the manageable resource. For example, a manager may be interested  
1197 when a service receives a new message. The manageable resource for the service has to notify  
1198 the manager when the event occurs. The manageable resource needs to know which manager is  
1199 interested in which information and the address(es) of the manager in order to send a notification  
1200 message whenever an event occurs.

1201 **Recommendation:** Use the WS-Notifications specification [WS-Notification].

## 1202 **Versioning**

1203 Version is an attribute of a resource identifying a set of supported capabilities and a sequence of  
1204 modifications to the component. There are two kinds of versioning; the resource version and the  
1205 Web service version. The format of the former is out of scope of this work. Version information  
1206 is useful so that the manager can know if the manageable Web Service interfaces have changed  
1207 since she obtained her copy.

1208 **Motivation:** A manager of manageable resources must have the ability to query the available  
1209 endpoint revisions, along with the corresponding change descriptions allowing that the manager  
1210 to discern the most appropriate and compatible interface of a particular manageability function  
1211 that her management client can use.

1212 **Recommendation:** Addressed by the WSDM TC as part of the “Management of Web Services”  
1213 (MOWS) specification under development for a Web service. W3C recommendations on this  
1214 topic [W3C TAG finding on Versioning] must be considered and accomodated. MUWS must  
1215 provide access to version information as part of the description of the identity property of a  
1216 manageable resource.

## 1217 **Security**

1218 There are many ways to categorize information security, but the most common today are  
1219 Confidentiality, Integrity, and Authentication. Additional concepts that can be arguably kept  
1220 separate are: Access Control, Non-repudiation, Availability, and Privacy. [see Glossary TBD]  
1221 Security requirements within the scope of manageability are not unique to manageability. Every  
1222 manageability endpoint and many business, or functional, endpoints have similar requirements  
1223 for confidentiality, integrity, and authentication, as well as for access control, availability, and  
1224 privacy (see the glossary definition of Security).

1225 **Motivation:** Resources have to be manageable in a secure way (see the glossary definition of  
1226 Security). Security infrastructure mechanisms should be composed and layered on top of  
1227 manageability exposed via Web services, similar to securing any other capability of a resource  
1228 exposed via a Web service. For example, access to a manageability operation can be granted to  
1229 only clients that present “manager’s identity” in a request message.

1230 **Recommendation:** WSDM will follow the recommendation of the OASIS WS-Security TC. For  
1231 the 1.0 specifications, WSDM should examine WS-Security in order to ensure nothing in the  
1232 specification precludes the composability of Security. In addition, security should be manageable  
1233 via Web services.

## 1234 **Registration and Discovery**

1235 Registration is a method of advertising the existence of an element so that it can be discovered.  
1236 Discovery is a method of locating an existing element so that it can be used or operated.  
1237 Discovery can be based on selection criteria or simply a name or identity of an element. Location  
1238 is a method of obtaining an address of an element. In the Web services sense, registration,  
1239 discovery and location can be represented by a set of operations and schema which may be  
1240 implemented by a Registry.

1241 **Motivation:** Manageable resources have to be discoverable by the managers. Manageable  
1242 resources exposed via Web services can be registered, discovered and located via a Registry.

1243 **Recommendation:** UDDI specification will satisfy most requirements. Existing Web services  
1244 discovery practices are sufficient.

## 1245 **Future Focus**

1246 Future versions of the WSDM TC specifications may also require support from the Web services  
1247 platform, as follows.

## 1248 Policy

1249 A Policy is a course of action, a guiding principle, or a procedure that is considered expedient,  
1250 prudent, or advantageous, for a given condition or event. A Policy describes a broad range of  
1251 service requirements, preferences, and capabilities.

1252 There are two kinds of policy governing the management of manageable resources. One policy  
1253 set describes how the client of a manageable resource interacts with the functional interfaces of  
1254 the resource. An example might be a policy describing the privacy of data., A policy set  
1255 describing how the manager of a manageable resource establishes operational requirements for  
1256 a resource. An example might be a policy describing service level agreements.

1257 **Motivation:** There are various policies that can be specified for manageable resources  
1258 including: authentication, access control, privacy, non-repudiation, service level agreement,  
1259 quality of service, routing, content inspection, auditing, and so on. MUWS must leverage  
1260 existing Web services specifications and technologies in its approach to applying policies to  
1261 manageable resources. MUWS should endorse a list of such specifications and technologies,  
1262 and should specify compatibility and interoperability requirements.

1263 **Recommendation:** None at this time.

## 1264 Name Resolution

1265 Name Resolution is necessary for management purposes and requires a name resolution  
1266 service. The name resolution service accepts a name identifier (URI) of a resource and returns  
1267 an address or reference to the manageability endpoint for the resource. The service should  
1268 return sufficient information to invoke the manageability endpoint. . The name resolution service  
1269 may be used to resolve names to references that are meaningful within other application  
1270 domains, and unrelated to management. One example would be service discovery.

1271 **Motivation:** A name resolution service is necessary for manageability because manageable  
1272 resources and manageability services expose many identifiers provided by existing  
1273 instrumentation and technologies. A manageability consumer needs the ability to get a  
1274 reference to any resource identifier, within any application domain, with which it wishes to  
1275 interact..

1276 **Recommendation:** None at this time.

## 1277 Transaction

1278 A “unit of work” that consists of multiple actions, typically an ordered set operation, that  
1279 is applied to a single resource, or applied to multiple resources. The “unit of work” should be  
1280 executed once and only once, even if, due to transmission failures or other errors, some request  
1281 within the transaction is received multiple times. There are three possible results from the  
1282 execution of a transaction:

- 1283 • all actions against all resources succeed
- 1284 • one or more actions fail, and all actions against all resources are rolled back.  
1285 Note: if roll back is not possible, or not supported, then the resource state should revert  
1286 to some known-as-good operational state or configuration.
- 1287 • one or more actions may fail, and the resource states are left as affected.

1288 **Motivation:** Grouping actions against resources and assuring their execution is of great value in  
1289 managing the resources. A manager may request that multiple actions or operations be  
1290 performed as a single “unit of work” preserving the integrity of the resources involved.

1291

1292 **Recommendation:** None at this time.

## 1293 **Flow**

1294 Flow, more often called workflow or workflow management, is the management of business  
1295 processes using information technology. When an organization defines, analyzes, and organizes  
1296 its resources and operations, workflow management systems can ensure the right information  
1297 reaches the right person, or computer application, at the right time. Business process  
1298 management (BPM) workflow or execution languages support the composing services into more  
1299 complex processes which may also be exposed as a Web service. The description of such  
1300 coordinated activities should include,, but should not be limited to:

1301 • constructs for the identification of partners

1302 • message correlation

1303 • fault detection and compensating activities

1304 • parallel and serial execution of services

1305 **Motivation:** Web services orchestration languages may be useful tools allowing Web services  
1306 management providers to support the more complex business oriented actions that can be taken  
1307 as a result of observations. A manageability interface may need to be defined for a business  
1308 process engine to properly and consistently monitor and control flows, and, for a composite Web  
1309 service that exposes the sessions and the state of a resource and any subordinate resource,.

1310 **Recommendation:** None at this time.

## 1311 **Negotiation**

1312 Negotiation is the process by which two services dynamically negotiate the terms of a contract.  
1313 A contract is negotiated by the initiators and the participants of that contract. A contract is a  
1314 document that represents a set of objectives assigned to a set of resources.

1315 **Motivation:** It is important for resources to understand what they can expect from other  
1316 resources, from managed resources, from management services, and from managers. The  
1317 process of negotiation enables initiators and participants to better describe their own level of  
1318 performance and how information shall be exchanged between the components of federated  
1319 deployments.

1320 **Recommendation:** None at this time.

1321