Web Service Management Framework - Foundation (WSMF-Foundation) Version 2.0

16 July 2003

Authors:

Nicolas Catania, Hewlett-Packard Company
Pankaj Kumar, Hewlett-Packard Company
Bryan Murray (editor), Hewlett-Packard Company
Homayoun Pourhedari, Hewlett-Packard Company
William Vambenepe, Hewlett-Packard Company
Klaus Wurster, Hewlett-Packard Company

Copyright Notice

Abstract

This specification describes how a manageable resource is exposed as a Web service. The Web Service Management Framework - Foundation (WSMF-Foundation) defines standard management interfaces for manageable resources as Web service operations. WSMF-Foundation describes how a manageable resource is discovered, how its management capabilities are described, how it is associated with other resources, and how to extend interfaces to address the management capabilities of resources in specific domains. This specification describes the foundation of the Web Services Management Framework. Additional specifications will describe applications of this framework to specific domains.

Status of this Document

The WSMF-Foundation specification is an initial public draft release and is provided for review and

evaluation only. Hewlett-Packard Company hopes to solicit your contributions and suggestions in the near future. Hewlett-Packard Company makes no warrantees or representations regarding the specification in any manner whatsoever.

Table of Contents

1. Introduction

As IT systems increase in complexity, it becomes increasingly important to manage these systems. It is desirable to be able to view and control the state associated with many resources within IT systems. A resource providing an interface to monitor and control the resource is a manageable resource. Different resources inevitably have different management interfaces.

One of the goals of Web services is to achieve universal interoperability between applications. The loosely coupled model of Web services provide for the flexible integration between heterogenous systems and applications. The flexible interoperability of Web services makes an excellent basis upon which to build standard management interfaces for IT resources.

The Web Service Management Framework (WSMF) makes use of Web services to provide the management interfaces to manageable resources. In this document the Web service providing the management interfaces is called a *managed object* , and the manageable resource associated with that Web service is called a *managed resource* . The extensibility mechanisms provided by WSMF-Foundation make it possible to design a managed object for any manageable resource.

Resources within an IT system are often related to each other in different ways. These relationships are important in providing insight into how a system will respond when a resource misbehaves or becomes unavailable. WSMF-Foundation represents these relationships in managed objects which provides a method for others to discover these relationships. The relationships also provide one means to discover other managed objects and resources in the system.

As resources proceed through their lifecycle or encounter difficulties, a managed object will inform others by sending notifications. WSMF-Foundation relies on an event system defined in [WS-Events] . Notifications allow an interested party to asynchronously keep track of a managed object's state and that of the underlying resource.

WSMF-Foundation defines a method for others to discover the management capabilities of a managed object. These capabilities are provided in a modular form to allow managed objects with different types of managed resources to expose different management interfaces. This modularity can also be used by a managed object to provide access control to its management capabilities.

WSMF builds upon the existing Web services framework. It uses Web Service Definition Language (WSDL) [WSDL] , and the XML schema specifications [XML Schema: Structures] and [XML Schema: Datatypes] to describe management interfaces and message structure. WSDL is particularly important because management interfaces are exposed as WSDL interfaces. [XPath] provides support for data manipulation. WSMF-Foundation is applicable to any version of SOAP ([SOAP 1.1] , [SOAP 1.2]) and its message processing model, as well as to other protocols that may be described in WSDL documents. A mapping to WSDL 1.1 [WSDL] is included in conjunction with this specification. However, as of this writing there is ongoing work in the definition of [WSDL 1.2] which may be useful in future versions of WSMF-Foundation. It is expected that a mapping of WSMF-Foundation to WSDL 1.2 will be made when that specification is released. Familiarity with these standards is important in the understanding of WSMF-Foundation.

In addition to layering on top of some XML standards, WSMF-Foundation is designed to make use of the existing Web service framework and its extensibility model. When mechanisms for security, reliability, routing, federation, and other aspects of the Web service framework become standardized, systems using WSMF-Foundation may incorporate these standards to provide a robust management framework tailored to the needs of the system.

2. Notations and Terminology

This section specifies the notations, namespaces, and terminology used in this specification.

2.1 Notational Conventions

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

This specification uses an informal syntax to describe the XML grammar of the messages 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.
- `<!-- extension elements -->` is a placeholder for elements from some other namespace (like `##other` in XML Schema).
- Characters are appended to attributes, elements, and `<!-- extension elements -->` 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 occurrance. 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 seperated 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.

Most applications using WSMF are expected to use SOAP bindings. For this reason, examples in this

document of complete messages are provided using the [SOAP 1.1] namespace. WSMF may be used with other versions of SOAP, or may be mapped to alternative bindings that do not use SOAP. The WSDL document associated with this specification includes a binding to SOAP over HTTP.

2.1.1 Namespaces

The [XML Namespace] URI that MUST be used by implementations of this specification is:

http://devresource.hp.com/drc/specifications/wsmf/2003/07/foundation/

The WSDL document describing WSMF-Foundation may be found at:

http://devresource.hp.com/drc/specifications/wsmf/2003/07/foundation/WSMF-Foundation.wsdl

A schema document containing definitions for some of the types for WSMF-Foundation may be found at:

http://devresource.hp.com/drc/specifications/wsmf/2003/07/foundation/WSMF-Foundation.xsd

The following namespaces are used in this document:

Prefix	Namespace
wsdl	http://schemas.xmlsoap.org/wsdl/
s	http://schemas.xmlsoap.org/soap/envelope/
xs	http://www.w3.org/2001/XMLSchema
evt	http://devresource.hp.com/drc/specifications/wsmf/2003/07/events/
wsmf	http://devresource.hp.com/drc/specifications/wsmf/2003/07/foundation/

Namespace names of the general form "http://example.org/..." and "http://example.com/..." represent application-dependent or context-dependent URI as defined in [IETF RFC 2396] .

2.2 Compliance

An implementation is not WSMF-Foundation compliant if it fails to satisfy one or more of the MUST or REQUIRED level requirements defined in this specification.

A message receiver MUST comply with this specification in order to process a WSMF-Foundation message successfully. A message sender MUST NOT use the **http://devresource.hp.com/drc/specifications/wsmf/2003/07/foundation/** [XML Namespace] identifier unless it is WSMF-Foundation compliant. A Web service provider MUST NOT use the **http://devresource.hp.com/drc/specifications/wsmf/2003/07/foundation/** XML Namespaces identifier in its WSDL description unless it is WSMF-Foundation compliant.

All parts of this specification are normative, with the exception of examples and sections explicitly marked as "Non-Normative". In cases where this document and the WSDL differ, the WSDL is considered normative.

2.3 Terminology

The following terms are described here and are used throughout this document and applications of this document to specific management domains:

Event

An *event* is a change in the state of a managed object.

Managed Object

A *managed object* is a management representation of a managed resource. A *managed object* implements one or more management interfaces to provide a means to monitor and/or control the underlying resource.

Managed Resource

A *managed resource* is a resource which is managed. A managed object exposes the management capabilities of a *managed resource* .

Management Interface

A *management interface* exposes a set of related management capabilities of a resource. A *management interface* is described by a WSDL interface (a portType in the case of WSDL 1.1) and implemented by a managed object.

Manager

A *manager* is a role which manages a managed object. A *manager* may be a management console or another managed object or some other software component. In turn, one *manager* may manage another *manager* by treating it as a managed object. Managed objects may have more than one *manager* . For instance, a managed object might have a management console as a *manager* and may have several other managed objects also acting as *manager* s.

Relation

A *relation* is a type of association between managed objects.

Relationship

A *relationship* specifies two managed objects and the relation describing how they are associated.

Resource

A *resource* is an entity that is a component of a deployed environment.

State

The *state* indicates where a resource currently is in its lifecycle.

3. Management Framework

WSMF-Foundation presents a framework for managing *resource* s through the use of Web services. This provides a platform-independent methodology for monitoring and controlling distributed resources. The management representation of a *resource* is called a *managed object* . In order to be managed, a *managed object* exposes one or more *management interface* s. These *management interface* s are accessed through Web services mechanisms and described in a [WSDL] document. This specification provides definitions for basic *management interface* s. Other documents will describe additional *management interface* s for applications of WSMF-Foundation to specific domains.

A *managed object* is managed through a set of *management interface* s, each of which provides access to a related set of management capabilities. Spreading the management capabilities across several *management interface* s offers a *managed object* the option of exposing a subset of its management capabilites to some of its *manager* s. It also allows the *managed object* to expose different management capabilities to different *manager* s based upon the privilege level of the *manager* . And it makes it possible to develop and deploy management capabilities over a period of time, exposing a subset of *management interface* s until full management capability is implemented and deployed.

The management framework consists of a universe of *managed object* s, one or more *manager* s, the relationships between *managed object* s, and the management capabilities exposed by *managed object* s: attributes, operations, and notifications. One *managed object* MAY provide *management interface* s for more than one *resource* , and more than one *managed object* MAY expose management capabilities for the same *resource* .

In the current WSDL 1.1 mapping, a *management interface* is described by a [WSDL] portType. A *managed object* MUST expose one or more *management interface* s, and thus will provide one or more WSDL portType definitions. *Management interface* s are exposed to *manager* s by defining a WSDL binding for a management portType and adding a port child element to a service element in a WSDL document. A *managed object* SHOULD support a SOAP over HTTP binding. If a SOAP over HTTP binding is supported for the interfaces described in this specification, the SOAP over HTTP binding specified in the WSDL document associated with this specification MUST be used. Every *management interface* MUST have its own WSDL interface. The attributes and operations (described below) making up the capabilities of a *management interface* are mapped to WSDL operations. If a *management interface* is exposed to a *manager* , all of the WSDL operations in the associated WSDL interface MUST be exposed to the *manager* .

Management interface s, and thus WSDL interfaces, are grouped together in [XML Namespace] to

describe the management capabilities for a class of *managed object* . Each class of *managed object* will use a different XML namespace. All of the WSDL interface definitions for a class will use the same XML namespace. WSMF-Foundation describes a single base class of *managed object* . Applications of WSMF to specific management domains will define new classes of *managed object* s using XML namespaces specific to that domain. Rather than re-defining all management capabilities, it is STRONGLY RECOMMENDED that applications of WSMF-Foundation extend from the collection of interfaces for a *managed object* defined by WSMF-Foundation by using the techniques described in **6. WSDL 1.1 Mapping** .

In addition to grouping WSDL interfaces into XML namespaces, a *management interface* is assigned to a category. A category encompasses a major subset of management capability. WSMF-Foundation defines six main categories: Configuration, Monitoring, Discovery, Control, Performance, and Security. It is RECOMMENDED that *management interface* s defined for new classes of *managed object* be assigned to one of these categories. This is most easily done by extending the base *managed object* portTypes. See **6. WSDL 1.1 Mapping** . When necessary, new categories may be defined for portions of the management capabilities for a new class of *managed object* . The base *managed object* does define WSDL portTypes that do not fall into these categories for the following purposes: *managed object* identity, collections of *managed object* s, and the event system.

A *management interface* provides access to a set of management capabilities. These capabilities are expressed as attributes and operations within the *management interface* . **Attribute** s are the properties exposed by the *managed object* allowing *manager* s to query and, in some cases, modify the current behavior of the *managed object* and the underlying *resource* . **Operation** s provide more complex access to and control of the *managed object* . Operations provide a *manager* a means to take action on the *managed object* to perform a specific action on it.

Managed object s express their association with other *managed object* s through *relationship* s. A *relationship* indicates another entity and describes how this *managed object* is associated with that other entity. A *managed object* has zero or more *relationship* s with other *managed object* s. Different types of associations are supported through different types of *relation* s. A *relation* is a type of association. It describes a one-way association from a *managed object* to another entity. A *relationship* includes the *relation* and adds the context of the *managed object* and the entity. WSMF-Foundation describes a small set of *relation* s. Applications of WSMF-Foundation to other management domains MUST describe the *relation* s between the new classes of *managed object* s for that management domain. See **3.3 Relationships** .

Notifications, as described by [WS-Events] , MAY be supported by a *managed object* . A *managed object* MAY implement the interfaces described in the WS-Events specification to generate events and to allow *manager* s to subscribe to events, query for events, and receive events. Extensions to the basic notification structure described in WS-Events are defined in WSMF-Foundation for the event types described in WSMF-Foundation. Applications of WSMF-Foundation to other domains MUST define their own extensions to the WS-Events notification structure for the events types needed in that management domain, if any.

Through the standard *management interface* s described by WSMF-Foundation, a *manager* can manage *managed object* s that originate from multiple operating systems and provided by multiple platform environments. A *managed object* MUST return a response, either successful or failure, in finite time for every request it receives from a *manager* . Using the protocols described in WSMF-Foundation, a *manager* MAY have management access, albeit limited, to *managed object* s on the other side of a firewall. Furthermore, a *managed object* exposing interfaces compliant with this specification can be managed by *manager* s implemented on different platforms and running under different operating systems, as long as the *manager* is also compliant with this specification.

WSMF-Foundation also describes how a WSDL document may be marked up to indicate which interfaces are *management interface* s, how to identify WSMF attributes, and how a *manager* can extract the desired management information from the document.

The following sections describe more about the representation of a *managed object* , how *managed object* s and their capabilities are discovered, relationships among *managed object* s, the event system used in WSMF-Foundation, the different states a *managed object* may be in, and common faults generated by WSMF-Foundation operations.

3.1 EntityReference

There are several operations in the *management interface* s and types which refer to a *managed object* . These references to *managed object* s MAY indicate either an actual *managed object* or another resource which may or may not be managed. It is useful to reference a non-managed entity to help a *manager* discover new resources, identity dependencies between resources, and to locate the root cause in cases of failure. A *managed object* reference is described in WSDL as an `EntityReference` .

An `EntityReference` has the following non-normative grammar:

```
<wsmf:EntityReference ...>
  <wsmf:ReferenceType>
      ManagedObject | ManagementEndpoint | OperationalWsdl |
      OperationalEndpoint | Undefined | Opaque
  </wsmf:ReferenceType>
  <wsmf:ReferenceUri>xs:anyURI</wsmf:ReferenceUri>
  <!-- extension elements -->
</wsmf:EntityReference>
```

- *wsmf:EntityReference*

  This element describes a reference to another entity as described in the `ReferenceType` element. It is RECOMMENDED that the preferred choice of reference type be the options in the order they are listed in the `ReferenceType` element.

- *wsmf:EntityReference/wsmf:ReferenceType*

  This element indicates the type of reference. `ManagedObject` specifies that the `ReferenceUri` element contains a URL indicating a concrete WSDL document describing the management interface for a *managed object* . `ManagementEndpoint` specifies the `ReferenceUri` element contains a URL to which management messages are sent. At least the `ManagedObjectIdentity` *management interface* MUST be implemented at this URL. `OperationalWsdl` specifies the `ReferenceUri` element contains a URL indicating the operational WSDL for a resource. The WSDL document at this location may or may not contain any *management interface* s. `OperationalEndpoint` specifies the `ReferenceUri` element contains a URL to which operational messages for the resource are sent. `Undefined` specifies that the `ReferenceUri` element identifies some other type of resource, such as a business process. It may or may not be possible to dereference the URI. It is RECOMMENDED that in this case, the URI use some means to convey information to a *manager* that may be useful in locating the resource if it has failed, perhaps including the machine name and strings identifying the resource. `Opaque` specifies that the `ReferenceUri` element contains an arbitrary URI that conveys no semantic meaning, such as a UUID.

- *wsmf:EntityReference/wsmf:ReferenceUri*

  This element contains the reference to another entity as described by the `ReferenceType` element.

- *wsmf:EntityReference/<!-- extension elements -->*

  An extensibility mechanism allows domain specific elements to be added to an entity reference. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

- *wsmf:EntityReference/@{any}*

  This is an extensibility mechanism to allow additional attributes to be specified. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

3.2 Discovery

There are two aspects of discovery that a *manager* will pursue with respect to *managed object* s. First, the *manager* needs to discover the resources it will manage and their *managed object* s. Here, the relationships a *managed object* has with other entities can help. Second, the *manager* needs to

discover the *management interface* s available on each of the *managed object* s it will be managing. Both topics are discussed in this section.

3.2.1 Discovery of *Managed Object* s

Manager s may discover or be informed of *managed object* s in many ways which fall outside the scope of this specification. Some of these include: command line arguments, configuration files, and UDDI. The method of *managed object* discovery specified here is through using the relationships one *managed object* has with another.

A snapshot of remote objects associated with a *managed object* is available by querying the *managed object* for its current set of *relationship* s. Each *relationship* identifies a remote object in its RelatedObject field. In many cases this field indicates another *managed object* . The *manager* can then discover the management capabilities of this *managed object* using the mechanism described in the next section. Depending upon the needs of the *manager* , it may then use this newly discovered *managed object* to discover additional *managed object* s using the same mechanism of querying for the *relationship* s. Some *managed object* s MAY have *relationship* s back to a previously discovered *managed object* so it is necessary to add protection from circular references when following *relationship* s. When using *relationship* s to discover new objects, it is best to start from a *managed object* that is at least a local root for a hierarchy. The reason is that not all *managed object* s will have reciprocal *relationship* s with each other.

Asking for the current *relationship* s gives a snapshot into the topology of a set of *managed object* s, but in many cases it is important to have a more dynamic view of how these relationships change. In order to achieve this dynamic view of *managed object* associations, it is necessary to use the event system exposed by the *managed object* s, if they support an event system. The event types important to tracking changes in *relationship* s is RelationshipAddedNotification and RelationshipRemovedNotification. See **3.4 Events** and [WS-Events] for more information.

3.2.2 Discovery of *Management Interface* s

There are two URLs that are important for a *managed object* . A *managed object* is identified by a URL to the concrete management WSDL which describes its *management interface* s. Management messages are sent to the URL(s) indicated in the management WSDL for the *management interface* s - this is often one URL for all *management interface* s, but it may be more than one. The URLs where management messages are sent are called "management endpoints".

3.2.2.1 Discovering Management WSDL URL From a Management Endpoint URL

A *managed object* MUST expose the *management interface* ManagedObjectIdentity. This interface contains the attribute `ManagementWsdlUrl` . This attribute returns the URL to the concrete management WSDL for the *managed object* . In cases where a *managed object* has different management endpoints for different *management interface* s, the *managed object* MUST expose the ManagedObjectIdentity at each of the management endpoints. If this is not done, a *manager* having a management endpoint may not be able to locate the management WSDL or the unique identity for the *managed object* . The `EntityReference` returned by the attribute `ManagedObjectReference` in the ManagedObjectIdentity *management interface* uniquely identifies the *managed object* .

3.2.2.2 Discovering Supported Management Interfaces From the Management WSDL URL

The task of determining the supported *management interface* s is primarily a matter of traversing the WSDL document by de-referencing the management WSDL URL. All *management interface* s MUST be included in a single WSDL "service" element. This allows a *manager* to stop traversing the WSDL document when it finds a "service" element containing a *management interface* . WSDL provides the ability to "include" other WSDL documents in the same XML namespace, and to "import" other WSDL documents having different XML namespaces. Also, a WSDL document may have more than one "service" element. For this reason, a *manager* MUST NOT assume that the "service" element found in the original WSDL URL will contain *management interface* s or that the XML namespace of the

management interface s is the same as the XML namespace of the initial WSDL document. Even if a "service element is found in this WSDL document, it may not contain any *management interface* s.

The RECOMMENDED order of WSDL traversal to locate the *management interface* s is to begin with the WSDL document indicated by the management WSDL URL. If at any point an element cannot be located in the current WSDL document, follow the "import" elements for WSDL documents in different namespaces.

1. Check all elements having a tag name of "service" in the WSDL document.
    1. Locate the "binding" element for each "port" child element.
    2. Skip "binding", and thus "port" elements for unsupported protocols/transports.
    3. Locate the "portType" element for the "binding" element.
    4. "portType" elements that include the attribute "baseManagementInterface" indicate *management interface* s.
2. Once all "service" elements in the initial WSDL document have been examined for *management interface* , follow all WSDL "import" links to discover any additional management information.

In order to make sure a *manager* can communicate with a *managed object* , the *manager* and *managed object* must agree on a common WSDL binding, and thus transport and message protocols. A SOAP over HTTP WSDL binding is provided with this specification. It is RECOMMENDED that implementations use this binding. Implementations MAY expose *management interface* s using other bindings. If a *managed object* does expose a SOAP over HTTP binding, it MUST use the binding in the WSDL document associated with this specification.

Once the "service" element containing the *management interface* s has been located, a *manager* can examine the associated "port" elements to determine the management endpoint for each *management interface* . A *manager* MUST NOT assume that all *management interface* s may be accessed at the same management endpoint. The "service" element containing *management interface* s MAY also contain non-management interfaces.

3.3 Relationships

In management systems, the relationships between *managed object* s are very important. It is also important to differentiate between types of relationships, as different *managers* may be interested in different types of relationships. For instance, a dependency relationship can be used by a *manager* to detemine which additional resources will be affected when one resource fails. Whereas, a duplication relationship, for example, can help determine which resources can replace one another. Relationships of one *managed object* can also help in discovering new *managed object* s. See section **3.2 Discovery** for more details.

A *relationship* is defined as two *managed object* s and the association, or *relation* , between them. A *relation* specifies only the association without the context of which objects are part of the association. Examples are: dependsOn, isDuplicatedBy, isTheSameColorAs, etc. A *relationship* adds the context of the participants in the association. A *relationship* is an ordered association. For example, "A dependsOn B", is not the same as "B dependsOn A". Think of the *relation* as a verb, one *managed object* as the subject, and the other *managed object* as the object of the *relationship* . WSMF-Foundation defines the "Relationship" type to represent a *relationship* . Since a *relationship* is acquired from one *managed object* , this *managed object* is implicitly considered the subject of the *relationship* . Both the *relation* and the object of the *relationship* are included as members of the "Relationship" type.

WSMF-Foundation defines a few basic types of *relation* s. Applications of WSMF-Foundation to other management domains, MUST specify *relation* s appropriate to that domain, if any, by defining the URI to indicate each *relation* and describing the semantics associated with that *relation* and the *managed object* s involved.

A *relationship* has the following non-normative grammar:

```
<wsmf:Relationship ...>
  <wsmf:HowRelated>wsmf:RelationType</wsmf:HowRelated>
  <wsmf:RelatedObject>wsmf:EntityReferenceType</wsmf:RelatedObject>
```

```
    <!-- extension elements -->
  </wsmf:Relationship>
```

- *wsmf:Relationship*

  This element specifies information about one relationship. A relationship is a type of association and the two objects sharing that association. In the case of a relationship instance, the first object is the *managed object* from which the relationship instance was obtained. Relationship instances are obtained by using the GetRelationships operation in the ManagedObject Discovery *management interface* described below.

- *wsmf:Relationship/wsmf:HowRelated*

  This element specifies the relation describing the type of association the *managed object* shares with another object. All values used for this field MUST be returned by the `SupportedRelations` attribute in the `ManagedObjectDiscovery` *management interface* . It is possible that the related object does not have a reciprocal relationship with this *managed object* . The type RelationType is a restriction of anyURI. The URI MUST be treated as an identifier of the type of relation. The parts of the URI MUST NOT be used to extract any meaning other than the type of association.

- *wsmf:Relationship/wsmf:RelatedObject*

  This element specifies another object with which this *managed object* shares some association. If possible, the related object SHOULD identify a *managed object* , however, the `EntityReferenceType` also provides a way to identify other objects. It is NOT REQUIRED for the related object to have a reciprocal relationship with this *managed object* .

- *wsmf:Relationship/<!-- extension elements -->*

  An extensibility mechanism allows domain specific elements to be added to a relationship. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

- *wsmf:Relationship/@{any}*

  This is an extensibility mechanism to allow additional attributes to be specified. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

WSMF-Foundation specifies only two relations, as a relation is specific to the types of *managed object* s and the domain in which they exist. See [WSMF-WSM] for an example. The relations defined by WSMF-Foundation are for the associations between a collection manager and a collection member, see **4.6 ManagedObject Collection Management Interface** for more information about collections. The relations are as follows:

- *http://devresource.hp.com/drc/specifications/wsmf/2003/07/relations/manager-of-collection/*

  This relation indicates the association a collection manager shares with each of the members of its collection. A collection manager MUST have a relationship with this relation with every member of its collection. The `RelatedObject` field of a `Relationship` MUST indicate the collection member.

- *http://devresource.hp.com/drc/specifications/wsmf/2003/07/relations/member-of-collection/*

  This relation indicates a *managed object* is a member of a collection. A *managed object* is RECOMMENDED to have a relationship with the `HowRelated` field of the `Relationship` set to this relation for every collection it knows it is a member. The `RelatedObject` field of the `Relationship` MUST indicate the collection manager.

3.4 Events

Events are an important part of any management system. This is also true in WSMF-Foundation. WSMF-Foundation provides an optional event system by adopting the event model and notification

system specified by [WS-Events] . The WS-Events specification describes how events can be propogated in a Web services environment using either or both a push and a pull mechanism. WSMF-Foundation specifies some specific event types which will support basic management. Applications of WSMF-Foundation to specific domains MUST specify additional event types needed to support management in those domains. See the [WS-Events] specification for definitions and descriptions of notifications, event types, and other related types.

WSMF-Foundation does not require that an event subsystem be supported by a *managed object* or a *manager* . However, if events will be supported by either a *managed object* or a *manager* , they MUST do it by following the [WS-Events] specification. WSMF-Foundation and its applications will specify the event types and notification extensions to be used in the case an event subsystem is supported. Additional event types MAY also be generated by a *managed object* . A *managed object* may choose to support the event push model or the event pull model or both event models as described in WS-Events. A *managed object* MAY consider the WS-Events WSDL interfaces it supports as *management interface* s even if those interfaces are not marked with the "baseManagementInterface" attribute described in section **6. WSDL 1.1 Mapping** . A *manager* SHOULD make the determination about whether the event interfaces belong to the management interfaces by examining the event types supported by the event interfaces. See [WS-Events] to find out more information about how to know which event types are supported.

A *manager* MAY choose to use any of the event models supported by the *managed object* from which it wishes to receive events. Section **3.2 Discovery** explains how a *manager* can discover which *management interface* s are supported by a *managed object* . A *manager* wishing to use a *managed object* 's event interfaces MUST be aware that the WS-Events interfaces MAY NOT be marked with the "baseManagementInterface" attribute described in section **6. WSDL 1.1 Mapping** .

Managed object s that support an event subsystem MUST set the `Source` element within the notifications they generate to the value of the `ReferenceUri` element of the `EntityReference` they return from the `ManagedObjectReference` attribute.

If a *managed object* does not support an event system, and a condition occurs under which an event would be generated, the *managed object* is NOT REQUIRED to take any action with regard to events. If the *managed object* supports either or both the push and pull event models and a condition occurs under which an event would be generated, the *managed object* MUST take the appropriate event related action as specified by the WS-Events, WSMF-Foundation, and WSMF-Foundation application specifications.

The remainder of this section describe extensions to the NotificationType defined in [WS-Events] . In these descriptions, only the extension elements and attributes are described. The following is an example of a message sent from *managed object* to a push subscriber when its State attribute is modified:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <evt:Notify xmlns:evt="http://devresource.hp.com/drc/specifications/wsmf/2003/07/events/"
                xmlns:wsmf="http://devresource.hp.com/drc/specifications/wsmf/2003/07/foundation/">
      <evt:NotificationList>
        <evt:Notification>
          <evt:Source>http://example.com/MO.wsdl</evt:Source>
          <evt:Type>
            http://devresource.hp.com/drc/specifications/wsmf/2003/07/managedobject/event/statechange
          </evt:Type>
          <evt:CreatedOn>2003-05-01T20:28:18Z</evt:CreatedOn>
          <evt:ExpiresOn>2003-05-02T20:28:18Z</evt:ExpiresOn>
          <wsmf:NewValue uri="
              http://devresource.hp.com/drc/specifications/wsmf/2003/07/managedobject/state/up/">
            </wsmf:State uri="http://example.com/State/ReadyToGo" />
          </wsmf:NewValue>
          <wsmf:OldValue uri="
              http://devresource.hp.com/drc/specifications/wsmf/2003/07/managedobject/state/down/" />
        </evt:Notification>
      </evt:NotificationList>
    </evt:Notify>
  </s:Body>
</s:Envelope>
```

3.4.1 RelationshipAdded

http://devresource.hp.com/drc/specifications/wsmf/2003/07/managedobject/event/relationship-added/

This notification MUST be generated by a *managed object* when it adds a relationship. It is RECOMMENDED that if several relationships are added at once, that all added relationships be included in the same notification. The new relationships MUST be included in the notification, as shown in the following non-normative grammar:

```
<wsmf:RelationshipAddedNotification>
  <!-- evt:NotificationType elements and attributes -->
  <wsmf:NewRelationshipList>
    <wsmf:Relationship> +
      wsmf:RelationshipType
    </wsmf:Relationship>
  </wsmf:NewRelationshipList>
  <!-- extension elements --> *
</wsmf:RelationshipAddedNotification>
```

- *wsmf:RelationshipAddedNotification*

  This type MAY be used wherever a Notification element may appear, as defined by [WS-Events] . It is generated by a *managed object* when it adds one or more new relationships.

- *wsmf:RelationshipAddedNotification/<!-- NotificationType ... -->*

  The standard Notification elements from [WS-Events] appear here.

- *wsmf:RelationshipAddedNotification/wsmf:NewRelationshipList*

  This element includes the list of relationships that were added. This list MUST contain at least one element, and MAY include multiple elements.

- *wsmf:RelationshipAddedNotification/<!-- extension elements -->*

  An extensibility mechanism to allow domain specific elements to be added to this notification. This allows a domain to define a special form of this notification with additional information. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

3.4.2 RelationshipRemoved

http://devresource.hp.com/drc/specifications/wsmf/2003/07/managedobject/event/relationship-removed/

This notification MUST be generated by a *managed object* when it removes a relationship. It is RECOMMENDED that if several relationships are removed at once, that all removed relationships be included in the same notification. The removed relationships MUST be included in the notification, as shown in the following non-normative grammar:

```
<wsmf:RelationshipRemovedNotification>
  <!-- evt:NotificationType elements and attributes -->
  <wsmf:RemovedRelationshipList>
    <wsmf:Relationship> +
      wsmf:RelationshipType
    </wsmf:Relationship>
  </wsmf:RemovedRelationshipList>
  <!-- extension elements --> *
</wsmf:RelationshipRemovedNotification>
```

- *wsmf:RelationshipRemovedNotification*

  This type MAY be used wherever a Notification element may appear, as defined by [WS-Events] . It is generated by a *managed object* when one or more of its relationships is removed.

- *wsmf:RelationshipRemovedNotification/<!-- NotificationType ... -->*

  The standard Notification elements from [WS-Events] appear here.

- *wsmf:RelationshipRemovedNotification/wsmf:RemovedRelationshipList*

  This element includes the list of relationships that were removed. This list MUST contain at least one element, and MAY include multiple elements.

- *wsmf:RelationshipRemovedNotification/<!-- extension elements -->*

  An extensibility mechanism to allow domain specific elements to be added to this notification. This allows a domain to define a special form of this notification with additional information. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

3.4.3 StateChanged

http://devresource.hp.com/drc/specifications/wsmf/2003/07/managedobject/event/state-changed/

This notification MUST be generated when a *managed object* modifies its State attribute. The notification is fired as the state is modified in order to have access to both the old value of state, and the new value of state. The format of the notification is shown in the following non-normative grammar:

```
<wsmf:StateChangedNotification>
  <!-- evt:NotificationType elements and attributes -->
  <wsmf:NewState>wsmf:StateType</wsmf:NewState>
  <wsmf:OldState>wsmf:StateType</wsmf:OldState>
  <!-- extension elements --> *
</wsmf:StateChangedNotification>
```

- *wsmf:StateChangedNotification*

  This type MAY be used wherever a Notification element may appear, as defined by [WS-Events] . It is generated by a *managed object* when its State attribute is modified either as a natural transition of its underlying resource, or direcly through the `SetState` operation.

- *wsmf:StateChangedNotification/<!-- NotificationType ... -->*

  The standard Notification elements from [WS-Events] appear here.

- *wsmf:StateChangedNotification/wsmf:NewState*

  This element specifies the value of the State attribute the *managed object* is transitioning to. The value is a `State` element representing the state and sub-states which describe the single *managed object* state being transitioned to. A *managed object* MUST include at least one `State` element in this list.

- *wsmf:StateChangedNotification/wsmf:OldState*

  This element specifies the value of the State attribute the *managed object* is transitioning from. The value is a `State` element representing the state and sub-states which describe the single *managed object* state being transitioned from. A *managed object* MUST include at least one `State` element in this list.

- *wsmf:StateChangedNotification/<!-- extension elements -->*

  An extensibility mechanism to allow domain specific elements to be added to this notification. This allows a domain to define a special form of this notification with additional information. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

3.4.4 AttributeValueChanged

http://devresource.hp.com/drc/specifications/wsmf/2003/07/managedobject/event/attribute-value-changed/

A *managed object* generates this notification when one of its attributes is modified either through normal operation or by a *manager* explicitly calling a "set" operation. The attributes included in this notification MUST follow the same access control policies as a query for the same attribute. Thus, if the *managed object* would normally not provide the value of the attribute to the querying *manager* , it should not include the attribute in a notification to the same *manager* . It is RECOMMENDED that a *managed object* group several attributes that have changed into a single notification. The format of the notification

has the following non-normative grammar:

```
<wsmf:AttributeValueChangedNotification>
  <!-- evt:NotificationType elements and attributes -->
  <wsmf:ChangedAttributeList>
    <wsmf:Attribute> +
      <wsmf:Interface>xs:QName</wsmf:Interface>
      <wsmf:Name>xs:NMTOKEN</wsmf:Name>
      <wsmf:NewValue>xml</wsmf:NewValue>
      <wsmf:OldValue>xml</wsmf:OldValue>
    </wsmf:Attribute>
  </wsmf:ChangedAttributeList>
  <!-- extension elements --> *
</wsmf:AttributeValueChangedNotification>
```

- *wsmf:AttributeValueChangedNotification*

  This type MAY be used wherever a Notification element may appear, as defined by [WS-Events] . It is generated by a *managed object* when one of its attributes is modified.

- *wsmf:AttributeValueChangedNotification/<!-- NotificationType ... -->*

  The standard Notification elements from [WS-Events] appear here.

- *wsmf:AttributeValueChangedNotification/wsmf:ChangedAttributeList*

  This element includes the list of attributes that have been modified. This list MUST contain at least one element, and MAY include multiple elements.

- *wsmf:AttributeValueChangedNotification/wsmf:ChangedAttributeList/wsmf:Attribute*

  This element specifies which attribute has been modified, its old value, and its new value.

- *wsmf:AttributeValueChangedNotification/wsmf:ChangedAttributeList/wsmf:Attribute/wsmf:Interface*

  This element specifies which *management interface* the modified attribute is found.

- *wsmf:AttributeValueChangedNotification/wsmf:ChangedAttributeList/wsmf:Attribute/wsmf:Name*

  This element specifies the name of the modified attribute.

- *wsmf:AttributeValueChangedNotification/wsmf:ChangedAttributeList/wsmf:Attribute/wsmf:OldValue*

  This element contains the previous value of the attribute. The contents MUST be the same as the contents of the response element for a query for this attribute when it has this value.

- *wsmf:AttributeValueChangedNotification/wsmf:ChangedAttributeList/wsmf:Attribute/wsmf:NewValue*

  This element contains the previous value of the attribute. The contents MUST be the same as the contents of the response element for a query for this attribute when it has this value.

- *wsmf:AttributeValueChangedNotification/<!-- extension elements -->*

  An extensibility mechanism to allow domain specific elements to be added to this notification. This allows a domain to define a special form of this notification with additional information. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

3.4.5 CollectionMemberAdded

http://devresource.hp.com/drc/specifications/wsmf/2003/07/managedobject/event/collection-member-added/

A *managed object* that exposes the `ManagedObjectCollection` *management interface* MUST generate this notification to a *manager* when a member is added to its collection. It is RECOMMENDED that a *managed object* group several members that have been added into a single notification. The format of the notification has the following non-normative grammar:

```
<wsmf:CollectionMemberAddedNotification>
  <!-- evt:NotificationType elements and attributes -->
  <wsmf:AddedMemberList>
    <wsmf:EntityReference>wsmf:EntityReferenceType</wsmf:EntityReference> +
  </wsmf:AddedMemberList>
```

```
  <!-- extension elements --> *
</wsmf:CollectionMemberAddedNotification>
```

- *wsmf:CollectionMemberAddedNotification*

  This type MAY be used wherever a Notification element may appear, as defined by [WS-Events] . It is generated by a *managed object* when a new member is added to its collection.
- *wsmf:CollectionMemberAddedNotification/<!-- NotificationType ... -->*

  The standard Notification elements from [WS-Events] appear here.
- *wsmf:CollectionMemberAddedNotification/wsmf:AddedMemberList*

  This element includes the list of collection members that were added. This list MUST contain at least one element, and MAY include multiple elements.
- *wsmf:CollectionMemberAddedNotification/wsmf:AddedMemberList/wsmf:EntityReference*

  This element is an EntityReference and specifies a member that was added to the *managed object* 's collection. The referenceType attribute of this EntityReference MUST be set to `ManagedObject` .
- *wsmf:CollectionMemberAddedNotification/<!-- extension elements -->*

  An extensibility mechanism to allow domain specific elements to be added to this notification. This allows a domain to define a special form of this notification with additional information. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

3.4.6 CollectionMemberRemoved

http://devresource.hp.com/drc/specifications/wsmf/2003/07/managedobject/event/collection-member-removed/

A *managed object* that exposes the `ManagedObjectCollection` *management interface* MUST generate this notification to a *manager* when a member is removed from its collection. It is RECOMMENDED that a *managed object* group several members that have been removed into a single notification. The format of the notification has the following non-normative grammar:

```
<wsmf:CollectionMemberRemovedNotification>
  <!-- evt:NotificationType elements and attributes -->
  <wsmf:RemovedMemberList>
    <wsmf:EntityReference>wsmf:EntityReferenceType</wsmf:EntityReference> +
  </wsmf:RemovedMemberList>
  <!-- extension elements --> *
</wsmf:CollectionMemberRemovedNotification>
```

- *wsmf:CollectionMemberRemovedNotification*

  This type MAY be used wherever a Notification element may appear, as defined by [WS-Events] . It is generated by a *managed object* when one of the members of its collection is removed.
- *wsmf:CollectionMemberRemovedNotification/<!-- NotificationType ... -->*

  The standard Notification elements from [WS-Events] appear here.
- *wsmf:CollectionMemberRemovedNotification/wsmf:RemovedMemberList*

  This element includes the list of collection members that were removed. This list MUST contain at least one element, and MAY include multiple elements.
- *wsmf:CollectionMemberRemovedNotification/wsmf:RemovedMemberList/wsmf:EntityReference*

  This element is an EntityReference and specifies a member that was removed from the *managed object* 's collection. The referenceType attribute of this EntityReference MUST be set to `ManagedObject` .
- *wsmf:CollectionMemberRemovedNotification/<!-- extension elements -->*

An extensibility mechanism to allow domain specific elements to be added to this notification. This allows a domain to define a special form of this notification with additional information. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

3.5 State

The lifecycle of a *managed resource* is important to a *manager* . WSMF-Foundation introduces an attribute named "State" to help a *manager* track a resource through its lifecycle. The State attribute provides for a list of state information where successive elements in the list provide more detailed values of state than the previous elements. The elements in the list can be thought of as state, sub-state, sub-sub-state, etc. A *managed object* MUST have at least one element to represent the state. Some values for state and sub-state are defined by WSMF-Foundation which are available for all *managed object* s to use to describe where their underlying resources are in their lifecycle. Additional values for State MAY be defined in specifications which describe applications of WSMF-Foundation to other domains. See the specification [WSMF-WSM] for new values of State defined for Web service *managed resource* s.

All values used for State MUST be defined and fully described in either the WSMF-Foundation specification, or an application of WSMF-Foundation to a specific domain. A *managed object* MUST NOT use values for State that have not been defined in a specification.

A *managed object* detecting a change in the state of its underlying resource MUST reflect this state change in its State attribute. A *managed object* supporting notifications MUST generate a `StateChanged` notification when it modifies its State attribute.

The following values MUST be used for the first element of the list of states returned from a query for a *managed object* 's State attribute. Subsequent elements of the list of states indicate sub-states and sub-sub-states, and all together describe the current state of the *managed object* . Applications of WSMF-Foundation SHOULD define values for sub-state appropriate to the *managed object* s and the domain.

- *http://devresource.hp.com/drc/specifications/wsmf/2003/07/managedobject/state/up/*

  This value of State indicates the underlying resource of this *managed object* is accepting and processing operational requests. The resource is available and healthy. Actions appropriate to the normal operation of the resource may be taken. All management capabilities supported by the *managed object* are available.

- *http://devresource.hp.com/drc/specifications/wsmf/2003/07/managedobject/state/down/*

  This value of State indicates the underlying resource of this *managed object* is not accepting any operational requests. The resource may not be healthy. Some management capabilities supported by the *managed object* may not be available.

- *http://devresource.hp.com/drc/specifications/wsmf/2003/07/managedobject/state/unknown/*

  This value of State indicates the *managed object* cannot determine the current state of the underlying resource. It is not likely that the resource can accept operational requests. The behavior of management operations is not guaranteed.

3.6 Faults

A *manager* may encounter errors when accessing the capabilities exposed from a *management interface* . These errors may be generated by the transport layer, the messaging layer, or may be detected in the management protocol. A *manager* MUST be prepared to handle all types of errors. Transport errors are implementation specific and are not further defined in this specification.

The `WsmfFaultDetail` element described here applies to SOAP bindings. A *managed object* that does not provide any SOAP bindings MAY ignore the `WsmfFaultDetail` element described here. However, all *managed object* s SHOULD use the values described below for `ErrorCode` if there is a mechanism to return an error code in the binding. A *managed object* returning a SOAP fault SHOULD include the `WsmfFaultDetail` element in the `detail` element of the SOAP Fault. The `WsmfFaultDetail` element is

described in the following non-normative grammar:

```
<wsmf:WsmfFaultDetail ...>
  <wsmf:ErrorList>
    <wsmf:Error ...> +
      <wsmf:ErrorCode>wsmf:ErrorCodeType</wsmf:ErrorCode>
      <wsmf:Message xml:lang="xs:language"?>xs:string</wsmf:Message> ?
      <!-- extension elements -->
    <wsmf:Error>
  <wsmf:ErrorList>
  <!-- extension elements -->
</wsmf:WsmfFaultDetail>
```

- *wsmf:WsmfFaultDetail*

  This element is the wrapper for all WSMF error information in a SOAP fault.

- *wsmf:WsmfFaultDetail/wsmf:ErrorList*

  This element includes the list of `Error` elements corresponding to the fault. This element MUST have at least one child `Error` element. An implementation MAY include multiple `Error` element if it detected more than one error while processing the message or if errors were detected by different modules in the implementation.

- *wsmf:WsmfFaultDetail/wsmf:ErrorList/wsmf:Error*

  This element encapsulates information about a single WSMF error.

- *wsmf:WsmfFaultDetail/wsmf:ErrorList/wsmf:Error/wsmf:ErrorCode*

  Indicate the type of the error. Several values for `ErrorCode` are specified in WSMF-Foundation. Additional values for `ErrorCode` will be specified in applications of WSMF-Foundation to additional domains. An implementation MUST NOT use values for `ErrorCode` that are not defined in either WSMF-Foundation or one of its applications to another domain. The type ErrorCodeType is a restriction from anyURI. The parts of the URI MUST NOT be used to extract any meaning other than the type of error.

- *wsmf:WsmfFaultDetail/wsmf:ErrorList/wsmf:Error/wsmf:Message*

  Optional element specifying implementation-dependent text describing the error. The optional attribute xml:lang SHOULD be used to indicate the language used for the text. If a *managed object* returns a text description of an error, it MUST use this element and MUST NOT add an extension element to specify the text description of the error.

- *wsmf:WsmfFaultDetail/wsmf:ErrorList/wsmf:Error/<!-- extension elements -->*

  Extension element used to add additional detailed information about an error.

- *wsmf:WsmfFaultDetail/<!-- extension elements -->*

  Extension element used to add additional detailed information about a WSMF SOAP fault.

WSMF-Foundation defines the following values for the `ErrorCode` element. Applications of WSMF-Foundation to other domains SHOULD define additional values for `ErrorCode` appropriate to that domain in the namespace used for that domain.

- *http://devresource.hp.com/drc/specifications/wsmf/2003/07/faults/malformed-message*

  This error code indicates that the message received by the *managed object* was malformed in some way. This error code MUST only be included in a SOAP Client Fault. A *manager* receiving this error MUST modify the request before resending it to the *managed object* .

- *http://devresource.hp.com/drc/specifications/wsmf/2003/07/faults/invalid-operation*

  This error code indicates that the operation specified by the request is not valid for this *managed object* . This fault MAY be returned for operations in *management interface* s that have not been implemented. This error code MUST only be included in a SOAP Client Fault. A *manager* receiving this error MUST modify the request before resending it to the *managed object* .

- *http://devresource.hp.com/drc/specifications/wsmf/2003/07/faults/missing-parameter*

This error code indicates that the request message is missing a required parameter for the operation specified. This error code MUST only be included in a SOAP Client Fault. A *manager* receiving this error MUST modify the request before resending it to the *managed object* .

- *http://devresource.hp.com/drc/specifications/wsmf/2003/07/faults/invalid-parameter*

  This error code indicates that the request message contains an invalid value for one of the parameters to the specified operation. This error code is used when a parameter value does not have the correct type, does not match a pattern specified for the parameter, is out of range, or some other invalid value. This error code MUST only be included in a SOAP Client Fault. A *manager* receiving this error MUST modify the request before resending it to the *managed object* .

- *http://devresource.hp.com/drc/specifications/wsmf/2003/07/faults/access-denied*

  This error code indicates that the *manager* does not have permission to access the specified operation. Authentication, authorization, or related security failures will cause this fault to be returned. This fault MAY be returned for access to *management interface* s that have not been exposed to the requester. A *managed object* MAY return the HTTP status, 403 Forbidden, instead of this SOAP fault from its SOAP HTTP binding. This error code MUST only be included in a SOAP Client Fault. A *manager* receiving this error MUST modify the request before resending it to the *managed object* .

- *http://devresource.hp.com/drc/specifications/wsmf/2003/07/faults/unavailable-attribute*

  This error code indicates that the operation specified by the request is attempting to access an attribute of the *managed object* that is not available. This error code MUST only be included in a SOAP Client Fault. A *manager* receiving this error MUST modify the request before resending it to the *managed object* .

- *http://devresource.hp.com/drc/specifications/wsmf/2003/07/faults/resource-exception*

  This error code indicates that the request successfully indicated an operation for the *managed object* , but the operation failed while executing. The failed operation indicates a problem with the underlying resource or the communication path to the resource. A *managed object* returning this value for an error code MUST set its State attribute to the value of Failed. This error code MUST only be included in a SOAP Server Fault.

- *http://devresource.hp.com/drc/specifications/wsmf/2003/07/faults/implementation-exception*

  This error code indicates that the *managed object* implementation generated an internal exception. Information about the exception MAY be included in additional child elements of the `Error` element. Care should be taken not to expose internal information about the implementation that could be used in security attack. A *managed object* returning this value for an error code SHOULD set its State attribute to the value of Failed. This error code MUST only be included in a SOAP Server Fault.

- *http://devresource.hp.com/drc/specifications/wsmf/2003/07/faults/not-implemented*

  This error code indicates that the *managed object* has not implemented the specified operation even though it is included in an exposed *management interface* . A future version of this *managed object* SHOULD include an implementation for this operation. This error code MUST only be included in a SOAP Server Fault.

The following is an example of using the `WsmfFaultDetail` element:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <s:Fault>
      <faultcode>s:Server</faultcode>
      <faultstring>Fault detected in managed object.</faultstring>
      <detail>
        <w:WsmfFaultDetail xmlns:w="http://devresource.hp.com/drc/specifications/wsmf/2003/07/foundat
          <w:ErrorList>
            <w:Error>
              <w:ErrorCode>
                http://devresource.hp.com/drc/specifications/wsmf/2003/07/faults/resource-exception
              </w:ErrorCode>
              <w:Message>The underlying resource is not responding.</w:Message>
            </w:Error>
```

```
          <w:ErrorList>
        </w:WsmfFaultDetail>
      </detail>
    </s:Fault>
  </s:Body>
</s:Envelope>
```

4. ManagedObject Interface Categories

This section describes several *management interface* s which provide the basis for all *managed object* s. These interfaces are useful for *managed object* s in most management domains. Each interface represents a category of management capability. This collection of interfaces are called a **ManagedObject** . A *managed object* either implements all operations defined for a particular interface, or none of the operations defined for that interface. It is not required to implement all categories of the *management interface* s described in this document. However, a *managed object* MUST implement the ManagedObjectIdentity *management interface* .

All of the *management interface* s in this section are implemented by a *managed object* and accessed from a *manager* . Each of the sections below describes a single category of *management interface* and it is described as a WSDL portType. The portType contains a set of message instances. The actual WSDL and XML Schema documents are referenced in an earlier section.

4.1 ManagedObject Identity *Management Interface*

All *managed object* s MUST implement the **ManagedObjectIdentity** *management interface* . It provides a method of finding other *management interface* s that the *managed object* exposes.

4.1.1 ManagedObjectReference Attribute

```
<wsmf:GetManagedObjectReference />

<wsmf:GetManagedObjectReferenceResponse>
  wsmf:EntityReferenceType
</wsmf:GetManagedObjectReferenceResponse>
```

- *wsmf:GetManagedObjectReference*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to query the *managed object* for its entity reference. The entity reference is the unique ID used to identify the *managed object* . The entity reference returned MUST have its `ReferenceType` field set to `ManagedObject` .

- *wsmf:GetManagedObjectReferenceResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to query for the *managed object* 's entity reference. The `ReferenceType` element in the returned `EntityReference` MUST be set to `ManagedObject` .

4.1.2 ManagementWsdlUrl Attribute

```
<wsmf:GetManagementWsdlUrl />

<wsmf:GetManagementWsdlUrlResponse>
  xs:anyURI
</wsmf:GetManagementWsdlUrlResponse>
```

- *wsmf:GetManagementWsdlUrl*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to query the *managed object* for the location of the WSDL document which

describes the management interfaces for the *managed object* .

- *wsmf:GetManagementWsdlUrlResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to query for the *managed object* 's management WSDL document.

4.2 ManagedObject Configuration *Management Interface*

The **ManagedObjectConfiguration** *management interface* is used by a *manager* to observe information about a *managed object* which is relatively static. Often, the information in this interface is set at deployment time and not changed during the lifetime of the *managed object* . Also, the information is usually the same every time the *managed object* is created, and may be the same on every deployment of the *managed object* .

4.2.1 Name Attribute

```
<wsmf:GetName />

<wsmf:GetNameResponse>
  xs:string
</wsmf:GetNameResponse>
```

- *wsmf:GetName*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to query the *managed object* for its name. The response message includes the name of the *managed object* . The name is intended for friendly recognition of the *managed object* by human beings and is not otherwise used in any WSMF-Foundation protocols. The name is initially specified by the implementer of the `ManagedObjectConfiguration` *management interface* or configured at deployment time. The name can be modified with the "SetName" operation in the `ManagedObjectControl` *management interface* .

- *wsmf:GetNameResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to query for the name of the *managed object* .

4.2.2 Type Attribute

```
<wsmf:GetType />

<wsmf:GetTypeResponse>
  wsmf:ManagedObjectType
</wsmf:GetTypeResponse>
```

- *wsmf:GetType*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to query the *managed object* for its type. The type `ManagedObjectType` is a restriction of xs:anyURI. The value specifies the type of the *managed object* . The Type is useful in specifying a class of *managed object* , of which there may be several instances in the system. This class offers a *manager* an idea about scope, intent, and behavior of the *managed object* and the resource it represents. The Type is not useful in determining the specific set of management capabilities available from the *managed object* , since both implementation and deployment may offer a subset of possible *management interface* s. To determine the specific set of capabilities exposed by the *managed object* , the management WSDL can be reviewed to discover the set of *management interface* s exposed by the object. This defines precisely the set of management capabilities available for this *managed object* .

- *wsmf:GetTypeResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to query for the type of a *managed object* . Implementations of WSMF-Foundation and any applications of WSMF-Foundation to specific domains MUST use the same value for Type when instances belong to the same class of *managed object* .

4.2.3 Description Attribute

```
<wsmf:GetDescription />

<wsmf:GetDescriptionResponse>
  xs:string
</wsmf:GetDescriptionResponse>
```

- *wsmf:GetDescription*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to query the *managed object* for its description. The description specifies human-readable information about the *managed object* . The description is intended for friendly recognition of the *managed object* by human beings. The description is not otherwise used in any WSMF-Foundation protocols. The description is either specified by the implementer of the ManagedObjectConfiguration *management interface* or configured by the person deploying the *managed object* .

- *wsmf:GetDescriptionResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to query for the *managed object* 's description. The value of the response is a string which specifies a description of the *managed object* .

4.2.4 Owner Attribute

```
<wsmf:GetOwner />

<wsmf:GetOwnerResponse>
  xs:string
</wsmf:GetOwnerResponse>
```

- *wsmf:GetOwner*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to query the *managed object* for its owner. The value of the response is a string which specifies the owner of the *managed object* . The owner indicates the person or company that has deployed the *managed object* implementation. This may be different from where the *managed object* is deployed in cases where a third-party hosts the deployment. The owner is not otherwise used in any WSMF-Foundation protocols.

- *wsmf:GetOwnerResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to query for the *managed object* 's owner.

4.2.5 Vendor Attribute

```
<wsmf:GetVendor />

<wsmf:GetVendorResponse>
  xs:string
</wsmf:GetVendorResponse>
```

- *wsmf:GetVendor*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to query the *managed object* for its vendor. The value of the response is a string which specifies the vendor of the *managed object* . The vendor indicates the developer of the *managed object* implementation. The vendor of the *managed object* MAY be different from the vendor of the underlying resource. The vendor is not otherwise used in any WSMF-Foundation protocols.

- *wsmf:GetVendorResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to query for the *managed object* 's vendor.

4.2.6 Version Attribute

```
<wsmf:GetVersion />

<wsmf:GetVersionResponse>
  xs:string
</wsmf:GetVersionResponse>
```

- *wsmf:GetVersion*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to query the *managed object* for the version of its underlying resource. The value of the response is a string which specifies the version of the *managed object* 's resource. The version is not otherwise used in any WSMF-Foundation protocols.

- *wsmf:GetVersionResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to query for the version of the underlying resource of a *managed object* .

4.2.7 ManagedObjectVersion Attribute

```
<wsmf:GetManagedObjectVersion />

<wsmf:GetManagedObjectVersionResponse>
  xs:string
</wsmf:GetManagedObjectVersionResponse>
```

- *wsmf:GetManagedObjectVersion*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to query the *managed object* for its version. The value of the response is a string which specifies the version of the *managed object* . This document makes no claims about the format of the version string for a *managed object* . The version is not otherwise used in any WSMF-Foundation protocols.

- *wsmf:GetManagedObjectVersionResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to query for the version of the *managed object* .

4.2.8 CreatedOn Attribute

```
<wsmf:GetCreatedOn />

<wsmf:GetCreatedOnResponse>
  xs:dateTime
</wsmf:GetCreatedOnResponse>
```

- *wsmf:GetCreatedOn*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to query the *managed object* for the date and time at which it began operating. The date and time the *managed object* begins operation MAY not be the same as the date and time the underlying resource begins operation. The *managed object* date and time is NOT REQUIRED to be later than the resource date and time.

- *wsmf:GetCreatedOnResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to query the *managed object* for the date and time at which it began operating. The value of the response has a type of XML Schema dateTime.

4.2.9 HostName Attribute

```
<wsmf:GetHostName />

<wsmf:GetHostNameResponse>
  xs:string
</wsmf:GetHostNameResponse>
```

- *wsmf:GetHostName*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to query the *managed object* for the host its underlying resource is running on. The value of the response is a string which specifies the host name of the *managed object* 's resource. The host name is not otherwise used in any WSMF-Foundation protocols.

- *wsmf:GetHostNameResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to query for the host name of the underlying resource of a *managed object* .

4.2.10 ManagedObjectHostName Attribute

```
<wsmf:GetManagedObjectHostName />

<wsmf:GetManagedObjectHostNameResponse>
  xs:string
</wsmf:GetManagedObjectHostNameResponse>
```

- *wsmf:GetManagedObjectHostName*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to query the *managed object* for the host it is running on. The value of the response is a string which specifies the host name of the *managed object* . The host name is not otherwise used in any WSMF-Foundation protocols.

- *wsmf:GetManagedObjectHostNameResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to query for the host name of the *managed object* .

4.3 ManagedObject Monitoring *Management Interface*

The **ManagedObjectMonitoring** *management interface* is used by a *manager* to observe information about a *managed object* which is likely to change.

4.3.1 State Attribute

```
<wsmf:GetState />

<wsmf:GetStateResponse>
  <wsmf:State uri="xs:anyURI">
    <wsmf:State uri="xs:anyURI" /> ?
  </wsmf:State>
</wsmf:GetStateResponse>
```

- *wsmf:GetState*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to query the *managed object* for its current state. A *managed object* MAY keep its overall state in terms of state, sub-state, sub-sub-state, etc. The response message will include the hiearchy of state values describing the overall state of the *managed object* . All values for state, sub-state, etc. returned by a *managed object* MUST also be included in the response to `GetSupportedStates` . A *managed object* MUST have at least a top-level value for its state, any sub-state values are optional. Values for state are defined in WSMF-Foundation. Values for sub-state, sub-sub-state, etc. will be defined in applications of WSMF-Foundation to specific domains.

- *wsmf:GetStateResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to query for the state of the *managed object* .

- *wsmf:GetStateResponse/wsmf:State*

  This element describes the top-level state of the *managed object* .

- *wsmf:GetStateResponse/wsmf:State/@uri*

  This attribute indicates the value of the state.

- *wsmf:GetStateResponse/wsmf:State/wsmf:State*

  This element describes the next-level state of the *managed object* . State levels may be nested arbitrarily deep - every `State` element MAY contain a child `State` element and MUST NOT contain more than one child `State` element. The attribute `uri` on each `State` element is required and indicates the value of that level of state.

4.3.2 SupportedStates Attribute

```
<wsmf:GetSupportedStates />

<wsmf:GetSupportedStatesResponse>
  <wsmf:StateList>
    <wsmf:State uri="xs:anyURI"> +
      <wsmf:State uri="xs:anyURI" /> *
    </wsmf:State>
  </wsmf:StateList>
</wsmf:GetSupportedStatesResponse>
```

- *wsmf:GetSupportedStates*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to query the *managed object* for the set of values it supports for the various levels of its State attribute.

- *wsmf:GetSupportedStatesResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to query for the set of values the *managed object* supports for its State attribute.

- *wsmf:GetSupportedStatesResponse/wsmf:StateList*

  This element is a wrapper for the list of top-level state values allowed. At least one child `State` element is REQUIRED in this list.

- *wsmf:GetSupportedStatesResponse/wsmf:StateList/wsmf:State*

  This element describes a top-level state of the *managed object* . Zero or more child `State` elements will describe sub-states for this top-level state.

- *wsmf:GetSupportedStatesResponse/wsmf:StateList/wsmf:State/@uri*

  This attribute indicates the value of the state.

- *wsmf:GetSupportedStatesResponse/wsmf:StateList/wsmf:State/wsmf:State*

  This element describes the next-level state of the *managed object* . State levels may be nested arbitrarily deep - every `State` element MAY contain zero or more child `State` elements. The child `State` elements describe possible sub-states for their parent `State` element. The attribute `uri` on each `State` element is required and indicates the value of that state.

4.4 ManagedObject Discovery *Management Interface*

The **ManagedObjectDiscovery** *management interface* is used by a *manager* to determine how this *managed object* is related to other *managed object* s. This is achieved through relationships between the current *managed object* and other *managed object* s which may not have been previously known to the *manager* . This interface is used by a *manager* when it first comes online, and when it determines that the set of relationships for the *managed object* has changed. If a *managed object* does not expose the `ManagedObjectDiscovery` interface, a *manager* MUST assume that this *managed object* has no relationships to other objects.

4.4.1 GetRelationships Operation

```
<wsmf:GetRelationships>
  <wsmf:RelationList> ?
    <wsmf:Relation>wsmf:RelationType</wsmf:Relation> +
  </wsmf:RelationList>
</wsmf:GetRelationships>

<wsmf:GetRelationshipsResponse>
  <wsmf:RelationshipList>
    <wsmf:Relationship>wsmf:RelationshipType</wsmf:Relationship> *
  </wsmf:RelationshipList>
</wsmf:GetRelationshipsResponse>
```

- *wsmf:GetRelationships*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to query the *managed object* for the set of relationships the *managed object* has with other entities. A *manager* MAY use the optional child element to indicate a list of relations in which it is interested. If this element is missing all relationships are returned, otherwise, only relationships with one of the specified relations are returned. The value of the response is a list of Relationship elements. The list may be empty which means the *managed object* currently has no relationships with other objects, or none with the specified relation if a list of relations is included in the request.

- *wsmf:GetRelationships/wsmf:RelationList*

  This optional element MAY be used by a *manager* to limit the returned list of relationships to only those with the relations specified in this list. A *manager* MAY omit this element which means all relationships will be returned. If this element is present, it MUST have at least one child `Relation` element.

- *wsmf:GetRelationships/wsmf:RelationList/wsmf:Relation*

  This element specifies a relation. The value is of type `RelationType` which is a restriction of the XML schema type `anyURI` .

- *wsmf:GetRelationshipsResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to query the *managed object* for

the set of relationships it currently has with other objects where the type of relation has the specified value.

- *wsmf:GetRelationshipsResponse/wsmf:RelationshipList*

  This element specifies encapsulates the list of relationships returned. This element MAY be empty.

- *wsmf:GetRelationshipsResponse/wsmf:RelationshipList/wsmf:Relationship*

  This element specifies information about one relationship. The relationship type is described in section **3.3 Relationships** . Zero or more occurances of this element are allowed.

4.4.2 SupportedRelations Attribute

```
<wsmf:GetSupportedRelations />

<wsmf:GetSupportedRelationsResponse>
  <wsmf:RelationList>
    <wsmf:Relation>wsmf:RelationType</wsmf:Relation> *
  </wsmf:RelationList>
</wsmf:GetSupportedRelationsResponse>
```

- *wsmf:GetSupportedRelations*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to query the *managed object* for the set of relations which the *managed object* can have with other entities. Relationships for this *managed object* MUST use relations from this set for all of its associations with other entities. The value of the response is a list of URIs which identify the relations that the *managed object* is uses to describe its associations with other entities. The list may be empty, meaning the *managed object* does not support any relations.

- *wsmf:GetSupportedRelationsResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to query for the *managed object* 's supported relations.

- *wsmf:GetSupportedRelationsResponse/wsmf:RelationList*

  This element specifies the list of relations supported by this *managed object* . This element MAY be empty.

- *wsmf:GetSupportedRelationsResponse/wsmf:RelationList/wsmf:Relation*

  This element specifies a type of association that is supported by the *managed object* . The type `RelationType` is a restriction of xs:anyURI. Zero or more occurances of this element are allowed.

4.5 ManagedObject Control *Management Interface*

The **ManagedObjectControl** *management interface* is used by a *manager* to modify the state of a *managed object* . The functionality in this interface is related to that provided in other interfaces, but is provided in an interface of its own to allow a *managed object* to expose access to information to a broad set of *manager* s, and to expose modification of this information to a smaller set of *manager* s.

4.5.1 State Attribute

```
<wsmf:SetState>
  <wsmf:State uri="xs:anyURI">
    <wsmf:State uri="xs:anyURI" /> ?
  </wsmf:State>
</wsmf:SetState>

<wsmf:SetStateResponse />
```

- *wsmf:SetState*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to modify the state of a *managed object* . Be aware that modifying the state of the *managed object* affects all *manager* s of the *managed object* . A *managed object* MUST generate all appropriate notifications associated with this state change. A *manager* must be aware that modifying the state manually with this attribute may not result in an actual change to the underlying resource. The resource may still be in a different state causing an immediate transition back to the original state with the associated notifications. Caution should be exercised in using this attribute to modify the state. As for all operations, *managed object* s are free to require credentials (through a mechanism not specified in this specification) to invoke this operation. It is expected that many *managed object* s will not allow most *manager* s, if any, to set the state of the *managed object* through this *management interface* . See the GetState operation in **4.3 ManagedObject Monitoring Management Interface** .

- *wsmf:SetState/wsmf:State*

  This element describes the top-level state of the *managed object* . This top-level state MAY have child `State` elements which indicate a sub-state.

- *wsmf:SetState/wsmf:State/@uri*

  This attribute indicates the value of the state.

- *wsmf:SetState/wsmf:State/wsmf:State*

  This element describes the next-level state of the *managed object* . State levels may be nested arbitrarily deep - every `State` element MAY contain a child `State` element and MUST NOT contain more than one child `State` element. The attribute `uri` on each `State` element is required and indicates the value of that level of state.

- *wsmf:SetStateResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to set the state for the *managed object* .

4.5.2 Name Attribute

```
<wsmf:SetName>
  <wsmf:Name>xs:string</wsmf:Name>
</wsmf:SetName>

<wsmf:SetNameResponse />
```

- *wsmf:SetName*

  This element specifies the request message of a request/response pair of messages. It is sent by a *manager* to modify the name of a *managed object* . An implementatin MAY persist this new value of `Name` so that it survives restarts, and it is NOT REQUIRED to do so. Be aware that modifying the name of the *managed object* affects all *manager* s of the *managed object* . See the GetName operation in the ManagedObjectConfiguration interface.

- *wsmf:SetName/wsmf:Name*

  This element specifies the new value which should be assigned to the *managed object* . The type of this element is a string. The value specifies the new name for the *managed object* .

- *wsmf:SetNameResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to set the name for the *managed object* .

4.6 ManagedObject Collection *Management Interface*

There are cases where a *manager* wants to take the same action on several *managed object* s. The

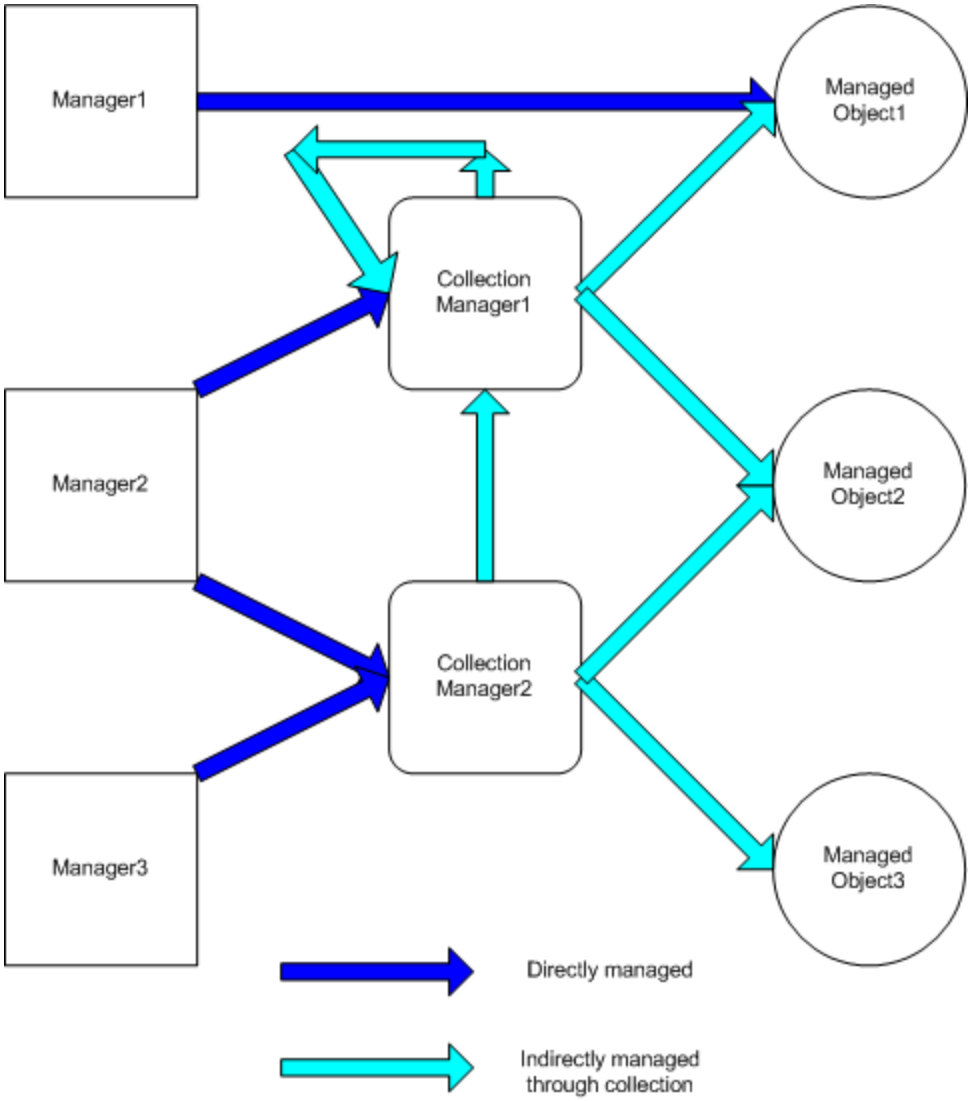
ManagedObjectCollection *management interface* allows one *managed object* to act as a proxy for several other *managed object* s. A **collection** is a set of *managed object* s that may be accessed through a `ManagedObjectCollection` *management interface* implemented by one *managed object* . A **collection manager** is a *managed object* that implements the `ManagedObjectCollection` *management interface* . A **collection member** is a *managed object* that may be addressed through a `ManagedObjectCollection` *management interface* of another *managed object* or itself, in addition to its own WSMF-compliant *management interface* s.

A *managed object* MAY be a *collection member* of zero or more *collection* s. A *collection member* is NOT REQUIRED to be aware that it is a member of a *collection* . A *managed object* that is a *collection member* of one or more *collection* s, MAY be accessed directly by zero or more *manager* s through any of its *management interface* s, and MAY be accessed through the *collection manager* for the *collection* s of which it is a member. How a *managed object* becomes a *collection member* of a particular *collection* is not defined in this specification.

A *managed object* MAY be a *collection manager* of zero or one *collection* s. A *collection manager* MAY consider itself as a *collection member* of the *collection* it controls, and is NOT REQUIRED to do so. A *collection manager* MAY include *managed object* s of any type in its *collection* . A *collection manager* MUST expose the `ManagedObjectCollection` *management interface* to *manager* s it allows to access its *collection* . A *collection manager* MAY be a *collection member* of another *collection* .

The `ManagedObjectCollection` *management interface* is used by a *manager* to perform the same action on zero or more *managed object* s that are *collection member* s of the *collection* . A *manager* can make a request to a *collection manager* , and the result will be that the same action is performed on all *collection member* s as long as they match the specified selection criteria. The *collection manager* MUST have a relationship with all of the *collection member* s in its *collection* , where the relation is "ManagerOfCollection". A *collection member* SHOULD have a relationship with all of its *collection manager* s, where the relation is "MemberOfCollection" and the related entity is the *collection manager* . See **3.3 Relationships** for more information about relationships. Actions taken by a *collection manager* MUST be reflected in the *collection member* s, such that another *manager* operating directly upon the *managed object* will see any changes made to the *managed object* by the *collection manager* . A query of an attribute MUST return the same result through a *collection manager* as a query directly to the *managed object* . A *collection manager* MUST NOT return from an action until all selected *collection member* s have either returned successfully, or have failed. Any requests made by a *collection manager* MUST complete in finite time.

The following picture and description help to understand the requirements associated with *collection* s:



- ManagedObject1, ManagedObject2, ManagedObject3, CollectionManager1, and CollectionManager2 are *managed object* s.
- The members of CollectionManager1's *collection* are: ManagedObject1, ManagedObject2, and CollectionManager1.
- The members of CollectionManager2's *collection* are: ManagedObject2, ManagedObject3, and CollectionManager1.
- CollectionManager1 considers itself a member of its *collection* , CollectionManager2 does not.
- CollectionManager2 has a *collection* as one of its *collection member* s.

The following sections will describe the `ManagedObjectCollection` *management interface* in more detail. Specifically, they will cover how to select a subset of *collection member* s to operate on for a given action, what is returned from requests made to a *collection* and how to determine which members were operated upon, and a description of the `ManagedObjectCollection` operations.

4.6.1 Select

Most of the operations in `ManagedObjectCollection` *management interface* have an optional "Select"

argument in their requests. This argument is used to identify a subset of *collection member* s to act upon. If the Select argument is not present in a request, all *collection member* s will be included in the action. The type of the Select argument is a string. The value of the Select argument MUST be an [XPath] expression identifying the *collection member* s to include in the action. The XPath expression operates on a virtual XML document (the document MAY not actually exist) with the following non-normative grammar:

```
<wsmf:ManagedObjectCollection ...>
  <wsmf:ManagedObject ...> *
    <wsmf:ManagedObjectReference>
      wsmf:EntityReferenceType
    </wsmf:ManagedObjectReference>
    <wsmf:InterfaceList>
      <wsmf:Interface ...> +
        <wsmf:Name>xs:QName</wsmf:Name>
        <wsmf:AttributeList> ?
          <!-- Attribute elements -->
        </wsmf:AttributeList>
        <!-- extension elements --> *
      </wsmf:Interface>
    </wsmf:InterfaceList>
    <!-- extension elements --> *
  </wsmf:ManagedObject>
  <!-- extension elements --> *
</wsmf:ManagedObjectCollection>
```

- *wsmf:ManagedObjectCollection*

  This is the root element of a virtual document that is used when the Select argument is specified in the Get, Set, or Invoke operations in the `ManagedObjectCollection` *management interface* s. This document is NOT REQUIRED to physically exist. The processing of the Select argument MUST behave the same as if the document does exist. The contents of this element is a list of ManagedObject elements. A *collection manager* MUST include a `ManagedObject` element for each *collection member* of its *collection* . This element MUST be empty only if there are no members in the collection.

- *wsmf:ManagedObjectCollection/wsmf:ManagedObject*

  This element describes the *management interface* s and attributes exposed by one *collection member* .

- *wsmf:ManagedObjectCollection/wsmf:ManagedObject/wsmf:ManagedObjectReference*

  This element references a *managed object* that is a member of the collection. This element is an EntityReference, and the ReferenceType child MUST have a value of `ManagedObject` .

- *wsmf:ManagedObjectCollection/wsmf:ManagedObject/wsmf:InterfaceList*

  This element is a wrapper for the list of *management interface* s exposed by the *managed object* . There MUST be an `Interface` element in this for every *management interface* exposed by the *managed object* to the *collection manager* .

- *wsmf:ManagedObjectCollection/wsmf:ManagedObject/wsmf:InterfaceList/wsmf:Interface*

  This element indicates that the associated *collection member* exposes this *management interface* .

- *wsmf:ManagedObjectCollection/wsmf:ManagedObject/wsmf:InterfaceList/wsmf:Interface/wsmf:Nam*

  This element specifies the name of the *management interface* . The value MUST be the QName of the WSDL portType describing the interface.

- *wsmf:ManagedObjectCollection/wsmf:ManagedObject/wsmf:InterfaceList/wsmf:Interface/wsmf:Attri*

  This element is a wrapper for the list of attributes available in this interface. This element has a child element for every attribute in the interface. Where "Get" operations exist for the attribute, the value of the attribute is also included. This element will not be present when no attributes are defined for the interface.

- *wsmf:ManagedObjectCollection/wsmf:ManagedObject/wsmf:InterfaceList/wsmf:Interface/wsmf:Attri*
  -- Attribute elements -->

An element will be present for every attribute in the interface.

```
<Owner>Widget Corporation</Owner>
<State
    uri="http://devresource.hp.com/drc/specifications/wsmf/2003/07/managedobject/state/down/">
  <wsmf:State uri=".../Inactive" />
</State>
```

- *wsmf:ManagedObjectCollection/wsmf:ManagedObject/wsmf:InterfaceList/wsmf:Interface/<!-- extension elements -->*

  An extensibility mechanism to allow domain specific elements to be added to the `Interface` element. This allows a domain to define a special form of this element with additional information. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

- *wsmf:ManagedObjectCollection/wsmf:ManagedObject/wsmf:InterfaceList/wsmf:Interface/@{any}*

  An extensibility mechanism to allow domain specific attributes to be added to the `Interface` element. This allows a domain to define a special form of this element with additional information. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

- *wsmf:ManagedObjectCollection/wsmf:ManagedObject/<!-- extension elements -->*

  An extensibility mechanism to allow domain specific elements to be added to the `ManagedObject` element. This allows a domain to define a special form of this element with additional information. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

- *wsmf:ManagedObjectCollection/wsmf:ManagedObject/@{any}*

  An extensibility mechanism to allow domain specific attributes to be added to the `ManagedObject` element. This allows a domain to define a special form of this element with additional information. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

- *wsmf:ManagedObjectCollection/<!-- extension elements -->*

  An extensibility mechanism to allow domain specific elements to be added to the `ManagedObjectCollection` element. This allows a domain to define a special form of this element with additional information. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

- *wsmf:ManagedObjectCollection/@{any}*

  An extensibility mechanism to allow domain specific attributes to be added to the `ManagedObjectCollection` element. This allows a domain to define a special form of this element with additional information. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

The following is a non-normative example of this virtual document for the collection CM2 from the diagram above.

```
<wsmf:ManagedObjectCollection
    xmlns:wsmf="http://devresource.hp.com/drc/specifications/wsmf/2003/07/foundation/">

  <wsmf:ManagedObject>
    <wsmf:ManagedObjectReference>
      <wsmf:ReferenceType>ManagedObject</wsmf:ReferenceType>
      <wsmf:ReferenceUri>http://example.com/MO2.wsdl</wsmf:ReferenceUri>
    </wsmf:ManagedObjectReference>
    <wsmf:InterfaceList>
      <wsmf:Interface>
```

```
      <wsmf:Name>wsmf:ManagedObjectIdentityInterface</wsmf:Name>
      <wsmf:AttributeList>
        <ManagedObjectReference>
          <wsmf:ReferenceType>ManagedObject</wsmf:ReferenceType>
          <wsmf:ReferenceUri>http://example.com/MO2.wsdl</wsmf:ReferenceUri>
        </ManagedObjectReference>
        <ManagementWsdlUrl>http://example.com/MO2.wsdl</ManagementWsdlUrl>
      </wsmf:AttributeList>
    </wsmf:Interface>
    <wsmf:Interface>
      <wsmf:Name>wsmf:ManagedObjectMonitoringInterface</wsmf:Name>
      <wsmf:AttributeList>
        <State
            uri="http://devresource.hp.com/drc/specifications/wsmf/2003/07/managedobject/state/down
          <wsmf:State uri=".../Inactive" />
        </State>
      </wsmf:AttributeList>
    </wsmf:Interface>
  </wsmf:InterfaceList>
</wsmf:ManagedObject>

<wsmf:ManagedObject>
  <wsmf:ManagedObjectReference>
    <wsmf:ReferenceType>ManagedObject</wsmf:ReferenceType>
    <wsmf:ReferenceUri>http://example.com/MO3.wsdl</wsmf:ReferenceUri>
  </wsmf:ManagedObjectReference>
  <wsmf:InterfaceList>
    <wsmf:Interface>
      <wsmf:Name>wsmf:ManagedObjectIdentityInterface</wsmf:Name>
      <wsmf:AttributeList>
        <ManagedObjectReference>
          <wsmf:ReferenceType>ManagedObject</wsmf:ReferenceType>
          <wsmf:ReferenceUri>http://example.com/MO3.wsdl</wsmf:ReferenceUri>
        </ManagedObjectReference>
        <ManagementWsdlUrl>http://example.com/MO3.wsdl</ManagementWsdlUrl>
      </wsmf:AttributeList>
    </wsmf:Interface>
    <wsmf:Interface>
      <wsmf:Name>wsmf:ManagedObjectMonitoringInterface</wsmf:Name>
      <wsmf:AttributeList>
        <State
            uri="http://devresource.hp.com/drc/specifications/wsmf/2003/07/managedobject/state/down
          <wsmf:State uri=".../Failed" />
        </State>
      </wsmf:AttributeList>
    </wsmf:Interface>
    <wsmf:Interface>
      <wsmf:Name>wsmf:ManagedObjectControlInterface</wsmf:Name>
      <wsmf:AttributeList>
        <Name />
        <State />
      </wsmf:AttributeList>
    </wsmf:Interface>
  </wsmf:InterfaceList>
</wsmf:ManagedObject>

<wsmf:ManagedObject>
  <wsmf:ManagedObjectReference>
    <wsmf:ReferenceType>ManagedObject</wsmf:ReferenceType>
    <wsmf:ReferenceUri>http://example.com/CM1.wsdl</wsmf:ReferenceUri>
  </wsmf:ManagedObjectReference>
  <wsmf:InterfaceList>
    <wsmf:Interface>
      <wsmf:Name>wsmf:ManagedObjectIdentityInterface</wsmf:Name>
      <wsmf:AttributeList>
        <ManagedObjectReference>
          <wsmf:ReferenceType>ManagedObject</wsmf:ReferenceType>
          <wsmf:ReferenceUri>http://example.com/CM1.wsdl</wsmf:ReferenceUri>
        </ManagedObjectReference>
        <ManagementWsdlUrl>http://example.com/CM1.wsdl</ManagementWsdlUrl>
      </wsmf:AttributeList>
    </wsmf:Interface>
    <wsmf:Interface>
      <wsmf:Name>wsmf:ManagedObjectMonitoringInterface</wsmf:Name>
      <wsmf:AttributeList>
        <State
            uri="http://devresource.hp.com/drc/specifications/wsmf/2003/07/managedobject/state/up/"
      </wsmf:AttributeList>
```

```
        </wsmf:Interface>
        <wsmf:Interface>
          <wsmf:Name>wsmf:ManagedObjectCollectionInterface</wsmf:Name>
          <wsmf:AttributeList>
            <Members>
              <Member>
                <wsmf:ManagedObjectReference>
                  <wsmf:ReferenceType>ManagedObject</wsmf:ReferenceType>
                  <wsmf:ReferenceUri>http://example.com/MO1.wsdl</wsmf:ReferenceUri>
                </wsmf:ManagedObjectReference>
              </Member>
              <Member>
                <wsmf:ManagedObjectReference>
                  <wsmf:ReferenceType>ManagedObject</wsmf:ReferenceType>
                  <wsmf:ReferenceUri>http://example.com/MO2.wsdl</wsmf:ReferenceUri>
                </wsmf:ManagedObjectReference>
              </Member>
              <Member>
                <wsmf:ManagedObjectReference>
                  <wsmf:ReferenceType>ManagedObject</wsmf:ReferenceType>
                  <wsmf:ReferenceUri>http://example.com/CM1.wsdl</wsmf:ReferenceUri>
                </wsmf:ManagedObjectReference>
              </Member>
            </Members>
          </wsmf:AttributeList>
        </wsmf:Interface>
      </wsmf:InterfaceList>
    </wsmf:ManagedObject>

  </wsmf:ManagedObjectCollection>
```

A restriction on the use of XPath for the Select argument to the Invoke operation is that the resulting node-set MUST include only `ManagedObject` nodes from the above virtual XML document.

The following examples show how the Select argument to the Invoke operation can choose subsets of *managed object* s to act upon. Every XPath processor has its own way of setting the namespace to be used in the XPath expression. It will be necessary to inform the XPath processor used that the prefix "wsmf" is mapped to the XML namespace "http://devresource.hp.com/drc/specifications/wsmf/2003/07/foundation/".

Select a *managed object* by the value of its management WSDL location:

```
Select =
  //wsmf:ManagedObject[
    wsmf:ManagedObjectReference/wsmf:ReferenceUri='http://example.com/MO2.wsdl']

Resulting managed objects from the virtual XML document above:
  MO2
```

Select all *collection manager* s:

```
Select =
  //wsmf:ManagedObject[
    wsmf:InterfaceList/wsmf:Interface/wsmf:Name='wsmf:ManagedObjectCollectionInterface']

Resulting managed objects from the virtual XML document above:
  CM1
```

Select all *managed object* s that have a State attribute indicating the resource is "Down":

```
Select =
  //wsmf:ManagedObject[
    wsmf:InterfaceList/wsmf:Interface/wsmf:AttributeList/State/@uri=
    'http://devresource.hp.com/drc/specifications/wsmf/2003/07/managedobject/state/down/']

Resulting managed objects from the virtual XML document above:
  MO2
  MO3
```

4.6.2 ManagedObjectResponseInformation

The result of the `Invoke` operation in the `ManagedObjectCollection` *management interface* is a list of elements indicating which *collection member* s were acted upon, and the results of that action. A *collection manager* MUST return a `ManagedObjectResponseInformation` element for each *collection member* selected in the `Invoke` operation. The `ManagedObjectResponseInformation` element has the

following non-normative grammar:

```
<wsmf:ManagedObjectResponseInformation ...>
  <wsmf:ManagedObjectReference>
    wsmf:EntityReferenceType
  </wsmf:ManagedObjectReference>
  <!-- response from invoking operation --> |
  <s:Fault>
    ...
  </s:Fault>
  <!-- extension elements --> *
</wsmf:ManagedObjectResponseInformation>
```

- *wsmf:ManagedObjectResponseInformation*

  This element encapsulates the response from `ManagedObjectCollection` operations Get, Set, and Invoke. In the case of Get, this element will contain either an InterfaceList element with the requested attribute values, or a Fault element. In the case of Set, this element will either be empty, or contain a Fault element. In the case of Invoke, this element will either contain the results of invoking the operation, be empty, or contain a Fault element.

- *wsmf:ManagedObjectResponseInformation/wsmf:ManagedObjectReference*

  This element indicates which *collection member* the information is for. The child `ReferenceType` element MUST have a value of `ManagedObject` .

- *wsmf:ManagedObjectResponseInformation/<!-- response from invoking operation -->*

  When the `ManagedObjectCollection` operation is Invoke, the result message is returned. This message may have an arbitrary namespace depending upon the specific method invoked.

- *wsmf:ManagedObjectResponseInformation/wsmf:Fault*

  This element is used to return fault information received from a *collection manager* from one of its members, to a *manager* . The fault information MAY include a SOAP `Fault` element if a SOAP binding is used, and it MAY include transport fault or other message protocol fault information. In the case of a SOAP fault, the contents of this element MUST be the SOAP `Fault` element received by the *collection manager* from its member.

- *wsmf:ManagedObjectResponseInformation/<!-- extension elements -->*

  An extensibility mechanism to allow domain specific elements to be added to the `ManagedObjectResponseInformation` element. This allows a domain to define a special form of this element with additional information. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

- *wsmf:ManagedObjectResponseInformation/@{any}*

  An extensibility mechanism to allow domain specific attributes to be added to the `ManagedObjectResponseInformation` element. This allows a domain to define a special form of this element with additional information. *Manager* s that understand these extensions MAY use them as long as they comply with all of the semantics associated with the extension. A *manager* that does not understand an extension MUST NOT process the extension.

4.6.3 Interface

The messages described by the `ManagedObjectCollection` *management interface* have the following non-normative grammar:

4.6.3.1 Invoke Operation

```
<wsmf:Invoke>
  <wsmf:Select>xs:string</wsmf:Select> ?
  <wsmf:Interface>xs:QName</wsmf:Interface>
  <wsmf:Name>xs:NMTOKEN</wsmf:Name>
  <wsmf:ArgumentList>  ?
    ...
```

```
  </wsmf:ArgumentList>
</wsmf:Invoke>

<wsmf:InvokeResponse>
  <wsmf:ManagedObjectResponseInformationList>
    <wsmf:ManagedObjectResponseInformation> *
      wsmf:ManagedObjectResponseInformationType
    </wsmf:ManagedObjectResponseInformation>
  </wsmf:ManagedObjectResponseInformationList>
</wsmf:InvokeResponse>
```

- *wsmf:Invoke*

  This element specifies the request message of a request/response pair of messages. This element contains several children elements describing the action to be performed. It is sent by a *manager* to a *collection manager* to invoke an operation in zero or more *collection member* s. The *collection manager* acts as a *manager* to zero or more other *collection member* s only in the sense that it is sending request messages to those *managed object* s. All response messages are packaged together and forwarded to the original requester and do not need further processing by the *collection manager* . A single invocation of the Invoke operation will cause the *collection manager* to iterate through all of the members of the collection testing whether it meets the specified criteria for inclusion in the command, and if so sends a request to that *managed object* and adds an element to the list of results to be returned to the original requester.

- *wsmf:Invoke/wsmf:Select*

  This optional element indicates the criteria by which other managed objects in the collection are included in the operation invocation. If the element is missing in the request, the invocation will be applied to all members of the collection. The Select element is described in detail in **4.6.1 Select** .

- *wsmf:Invoke/wsmf:Interface*

  This required element indicates in which interface the operation to be invoked exists. The type of this element is a QNAME. The value of the QNAME MUST identify the portType of the *management interface* in which the operation is defined.

- *wsmf:Invoke/wsmf:Name*

  This required element indicates the name of the operation to be invoked. The type of this element is string. The value MUST match the name of an operation in the specified *management interface* . A *managed object* MUST return a fault if the named operation does not exist in the specified interface.

- *wsmf:Invoke/wsmf:ArgumentList*

  This optional element specifies the arguments for the operation to be invoked. If the operation does not take any arguments, or no arguments are being sent to the operation, this element MUST NOT be present. The child elements of the `ArgumentList` element are a list of elements which will be sent as children elements of the specified operation element. The list content MUST be identical to the content that would be sent to a single *managed object* invocation of the specified oepration. For example, the Subscribe operation in one of the [WS-Events] interfaces has the following form:

  ```
  <evt:Subscribe>
    <evt:EventSelector>...</evt:EventSelector>
    <evt:ExpirationTime>...</evt:ExpirationTime>
    <evt:CallbackUrl>...</evt:CallbackUrl>
  </evt:Subscribe>
  ```

  The corresponding invocation using the Invoke operation is as follows:

  ```
  <wsmf:Invoke>
    <wsmf:Select>...</wsmf:Select>
    <wsmf:Interface>evt:EventPushPortType</wsmf:Interface>
    <wsmf:Name>Subscribe</wsmf:Name>
    <wsmf:ArgumentList>
      <evt:EventSelector>...</evt:EventSelector>
      <evt:ExpirationTime>...</evt:ExpirationTime>
      <evt:CallbackUrl>...</evt:CallbackUrl>
    </wsmf:ArgumentList>
  ```

```
  </evt:Invoke>
```

- *wsmf:InvokeResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to invoke an operation on a set of *managed object* s that are members of the collection. This element does not indicate that the request to every selected member of the collection was successful, only that no error was detected during the processing of this collection command and the iteration through the members. Failures encountered by the *collection manager* in sending requests to *collection member* s are included in the results sent back to the original requester. See section **4.6.2 ManagedObjectResponseInformation** .

- *wsmf:InvokeResponse/wsmf:ManagedObjectResponseInformationList*

  This element indicates the return value of the Perform operation.

- *wsmf:InvokeResponse/wsmf:ManagedObjectResponseInformationList/wsmf:ManagedObjectRespo*

  This element specifies a *managed object* which participated in the operation invocation. If there was no failure from the request to invoke the operation on the member and there is no return value from the operation, the `ManagedObjectResponseInformation` element will be empty, indicating only which *managed object* was included in the collection request. Otherwise, the `ManagedObjectResponseInformation` element will contain either the return value from the operation or a SOAP fault indicating a failure to invoke the operation. Zero or more occurrances of this element are allowed in the response. See section **4.6.2 ManagedObjectResponseInformation** for a description of the contents of this element. A *collection manager* MUST include one `ManagedObjectResponseInformation` element for every *collection member* matching the criteria specified in the Select argument in the request message.

The following example of a request message fragment shows how an attribute can be queried using the "Invoke" operation:

```
<wsmf:Invoke>
  <wsmf:Select>...</wsmf:Select>
  <wsmf:Interface>wsmf:ManagedObjectConfigurationInterface</wsmf:Interface>
  <wsmf:Name>GetVersion</wsmf:Name>
</evt:Invoke>
```

The response message to this query will contain a list of `ManagedObjectResponseInformation` elements, one for each member selected in the request. Each of the elements in the list will have a form similar to the following:

```
<wsmf:InvokeResponse>
  <wsmf:ManagedObjectResponseInformation>
    <wsmf:ManagedObjectReference>
      <wsmf:ReferenceType>ManagedObject</wsmf:ReferenceType>
      <wsmf:ReferenceUri>http://example.com/MO3.wsdl</wsmf:ReferenceUri>
    </wsmf:ManagedObjectReference>
    <wsmf:GetVersionResponse>
      1.12.527  June 30, 2003
    </wsmf:GetVersionResponse>
  </wsmf:ManagedObjectResponseInformation>
</wsmf:InvokeResponse>
```

The following example of a request message fragment shows how an attribute can be modified using the "Invoke" operation. A successful response will be a list of `ManagedObjectesponseInformation` elements containing only the `EntityReference` to the *managed object* .

```
<wsmf:Invoke>
  <wsmf:Select>...</wsmf:Select>
  <wsmf:Interface>wsmf:ManagedObjectControlInterface</wsmf:Interface>
  <wsmf:Name>SetName</wsmf:Name>
  <wsmf:ArgumentList>
    <wsmf:Name>MyWebService</wsmf:Name>
  </wsmf:ArgumentList>
</evt:Invoke>
```

And the resulting response would look like:

```
<wsmf:InvokeResponse>
  <wsmf:ManagedObjectResponseInformation>
```

```
    <wsmf:ManagedObjectReference>
      <wsmf:ReferenceType>ManagedObject</wsmf:ReferenceType>
      <wsmf:ReferenceUri>http://example.com/MO3.wsdl</wsmf:ReferenceUri>
    </wsmf:ManagedObjectReference>
    <wsmf:SetNameResponse />
  </wsmf:ManagedObjectResponseInformation>
</wsmf:InvokeResponse>
```

A response to the `Invoke` operation containing a fault from a SOAP over HTTP binding could look like:

```
<wsmf:InvokeResponse>
  <wsmf:ManagedObjectResponseInformation>
    <wsmf:ManagedObjectReference>
      <wsmf:ReferenceType>ManagedObject</wsmf:ReferenceType>
      <wsmf:ReferenceUri>http://example.com/MO3.wsdl</wsmf:ReferenceUri>
    </wsmf:ManagedObjectReference>
    <wsmf:Fault>
      <s:Fault>
        <faultcode>s:Server</faultcode>
        <faultstring>Fault detected in resource</faultstring>
        <faultactor>http://example.com/MO3</faultactor>
        <detail>
          <wsmf:wsmfFaultDetail>
            <wsmf:ErrorList>
              <Error>
                <ErrorCode>
                  http://devresource.hp.com/drc/specifications/wsmf/2003/07/faults/resource-exception
                </ErrorCode>
                <wsmf:Message>
                  The resource is not responding.
                </wsmf:Message>
              </Error>
            </wsmf:ErrorList>
          </wsmf:wsmfFaultDetail>
        </detail>
      </s:Fault>
    </wsmf:Fault>
  </wsmf:ManagedObjectResponseInformation>
</wsmf:InvokeResponse>
```

4.6.3.2 Members Attribute

```
<wsmf:GetMembers />

<wsmf:GetMembersResponse>
  <wsmf:EntityReferenceList>
    <wsmf:EntityReference>  *
      wsmf:EntityReferenceType
    </wsmf:EntityReference>
  </wsmf:EntityReferenceList>
</wsmf:GetMembersResponse>
```

- *wsmf:GetMembers*

  This element specifies the request message of a request/response pair of messages. This element has no child elements. It is sent by a *manager* to a *managed object* exposing the ManagedObjectCollection *mangement interface* to query for the members of the collection.

- *wsmf:GetMembersResponse*

  This element specifies the response message of a request/response pair of messages. It is received by a *manager* as an indication of a successful attempt to retrieve the members of the collection.

- *wsmf:GetMembersResponse/wsmf:EntityReferenceList*

  This element is a wrapper for the list of *collection member* s.

- *wsmf:GetMembersResponse/wsmf:EntityReferenceList/wsmf:EntityReference*

  This element specifies one member of the collection. Zero or more occurrances of this element are allowed in the response.

5. Applying WSMF to a Specific Domain

WSMF-Foundation provides basic management capabilities that are useful for managing any resource. However, many types of resources will need to expose management capabilities beyond those specified in this document. For this reason, an application of WSMF-Foundation can be done to provide for the management needs of any resource. This section will describe how to apply the base management interfaces described in this document to specific management domains. For an example of an application of WSMF-Foundation, see the specification [WSMF-WSM] which describes how to apply WSMF-Foundation to the Web services domain.

The mechanism for adding new management capabilities is to add a new *management interface* . The new capabilities are added by defining the attributes and operations to be included in the *management interface* . All WSMF compliant attributes MUST support query capability, and MAY support a modification capability. A designer MAY want to place the query portion of an attribute in a seperate *management interface* from the modification portion of the attribute to help facilitate interface-based access control.

When WSMF management capabilities are being defined for a new type of resource, it MAY be necessary to define a collection of *management interface* s. A collection of interfaces is used to represent all of the management capabilities of a new type of *managed object* . In addition to defining the *management interface* s, the event types, relations, states, and error codes needed for the resource domain beyond those described in WSMF-Foundation MUST be defined, described, and documented. The new application of WSMF SHOULD be documented in a specification similar to [WSMF-WSM] . The application is STRONGLY RECOMMENDED to use the management capabilities already defined in WSMF-Foundation, adding only new capabilities to newly defined *management interface* s.

A domain may require several new types of *managed object* s to represent the model for that domain. The *managed object* s may expose an existing model prevalent in that domain, or may expose a model designed specifically for WSMF. The relationships between the types of *managed object* s is important to model.

Important information to list in the documentation for the new types of *managed object* includes the URIs for any relations, states, notification types, and error codes needed for the new types of *managed object* . The URIs MUST be defined in a namespace appropriate for the domain.

6. WSDL 1.1 Mapping

Different types of resources are represented by different types of *managed object* s. Mapping a new type of *managed object* to WSDL 1.1 involves defining a new [XML Namespace] . Each type of *managed object* is defined in its own XML namespace. A *management interface* is mapped to a WSDL 1.1 portType in the new namespace. Management operations are mapped directly to WSDL operations. Management attributes are mapped to one or two WSDL operations: query capability is mapped to one WSDL operation, and optional modification capability is mapped to another WSDL operation. The convention used in WSMF-Foundation for the attribute operation names is to add the prefix "Get" or "Set" to the name of the attribute. For instance, the attribute "Name" is accessed with the WSDL operations "GetName" and "SetName".

A new *management interface* MAY require one or more other *management interface* s (explained below) to be exposed, or it may be a new stand-alone interface if it does not require any other interfaces. An implementation of a *managed object* exposing a WSDL portType that requires other portTypes to be exposed MUST also expose all of the required portTypes. New portTypes MUST be defined in an XML namespace controlled by the designer and MUST NOT be defined in any WSMF XML namespace. As new interfaces become standardized and used by several *managed object* implementers, WSMF may incorporate the interface into a WSMF XML namespace. Implementers of new *management interface* s are encouraged to reuse types defined in WSMF WSDL and XML schemas where possible. New capabilites MUST NOT be defined in new interfaces if they are the same as capabilities already defined in WSMF interfaces. Instead the inheritance mechanism described below SHOULD be used to incorporate the WSMF defined capabilities. It is STRONGLY RECOMMENDED that new interfaces be defined using the major categories of: Configuration, Monitoring, Discovery, Control, Performance, and Security. These are the categories used by WSMF-Foundation and have been derived from OSI categories.

Notification types, relations, states, and error codes do not have any direct realization in a WSDL document. However, the designer MUST specify URI values for any new instances created for the new type of *managed object* .

6.1 baseManagementInterface WSDL Attribute

WSMF-Foundation defines a method to indicate an interface is a *management interface* and to allow one *management interface* to extend from another. Interfaces need to be marked as *management interface* s to improve the ability to discover the capabilities of a *managed object* . Inheritance helps promote the reuse of more generic management capabilities. A WSDL 1.1 portType is marked as a *management interface* by adding the XML attribute "wsmf:baseManagementInterface" to the portType element in a WSDL document. This attribute can have one of two values. If the portType requires other portTypes to also be implemented, the attribute value is a list where each item in the list is a QName. The value of the attribute identifies all of the other *management interface* s which MUST be implemented if this portType is implemented. If the portType does not require any other portTypes, the attribute value is the fixed string "#base", which indicates that it describes a stand-alone *management interface* . This attribute MUST appear on all WSDL portTypes describing *management interface* s. There is one exception to this rule: the interfaces described in [WS-Events] that an event producer implements are also considered *management interface* s. In order to use the WSDL supplied with the WS-Events specification without modification, it is permissable to omit the wsmf:baseManagementInterface for only these interfaces. The example following shows how this attribute is used:

```
<wsdl:portType name="ManagedObjectCollectionInterface"
      wsmf:baseManagementInterface="#base">
  ...
</wsdl:portType>

<wsdl:portType name="ManagedObjectMonitoringInterface"
      wsmf:baseManagementInterface="#base">
  ...
</wsdl:portType>

<wsdl:portType name="ServiceMonitoringInterface"
      wsmf:baseManagementInterface="wsmf:ManagedObjectMonitoringInterface">
  ...
</wsdl:portType>

<wsdl:portType name="ObjectServerInterface"
      wsmf:baseManagementInterface="wsmf:ManagedObjectMonitoringInterface
      wsmf:ManagedObjectCollectionInterface">
  ...
</wsdl:portType>
```

In cases where more than one base interface is required, and more than one of these interfaces also require the same base interface, the resulting interface is considered to have a single copy of the duplicate inherited interface. For example consider the following interface inheritance:

- B requires A
- C requires A
- D requires B and C

Even though interface A is required twice, only a single copy of interface A will be present in an implementation of interface D.

A *manager* MAY access any new *management interface* s it finds in the discovery process as long as it understands the interface and complies with all semantics associated with the interface. A *manager* MUST ignore a *management interface* if it does not recognize the interface or cannot comply with the semantics of the interface. A *manager* MAY invoke the query portions of an interface by accessing a *managed object* 's attributes without understanding the semantics of the interface, if the values are used only for display purposes. If a *management interface* is derived from another interface and a *manager* does not understand the derived interface, but does understand the base interface, the *manager* MAY use the base interface without using any features from the derived interface. A *manager* MUST NOT use an interface if it does not understand the base interface.

6.2 attributeType WSDL Attribute

WSMF defines another marker attribute to differentiate between WSDL operations that define WSMF operations versus WSDL operations which expose the query and modify capabilities of a WSMF attribute. *Managed object* attributes have a WSDL operation to query the value of the attribute. The name of this query operation will have the form "GetProperty", where "Property" is replaced with the name of the attribute. The request message for this operation takes no arguments. The response message returns the current value of the attribute for the *managed object* . The value returned MAY represent a cached value, but the value MUST be the current value of the attribute of the *managed object* . If the attribute may also be modified by the *manager* , the *managed object* will also expose a WSDL operation to modify the attribute. This modify operation MAY be in a different *management interface* from the query operation for purposes of access control. The name of the modify operation will have the form "SetProperty", where "Property" is the name of the attribute. The request message takes a single argument which is the new value of the attribute. The response message is empty. The type of the argument in the request message for the modify operation MUST have the same type as the type of the response to the query operation. The WSDL attribute used to mark the query and modify operations for an attribute is "wsmf:attributeType". This attribute MUST mark all query and modify operations associated with access to a *managed object* 's attributes as described here. The type of the "wsmf:attributeType" attribute is QName. The value MUST identify the type of the *managed object* attribute.

```
<wsdl:operation name="GetVersion"
    wsmf:attributeType="xs:string" ...>
  ...
</wsdl:operation>
```

6.3 relation WSDL attribute

WSMF-Foundation provides a way to retrieve all of a *managed object* 's relationships by using the `GetRelationships` operation. in this case, the response is a list of `Relationship` elements. There are cases where it is desirable to retrieve a list of *managed object* s that have a certain type of relationship. For these cases, WSMF-Foundation adds "short-cut" attributes. An example in the `ManagedObjectCollection` interface is the `Members` attribute. This attribute returns a list of *managed object* s where the *managed object* has a "ManagerOfCollection" relation with each member of the list. In order to indicate to a *manager* that this is a "short-cut" attribute returning *managed object* s with a certain relation, a WSDL attribute has been defined to mark this WSDL operation. The WSDL attribute "wsmf:relation" MUST be included in WSDL operation elements when the operation returns a subset of all relationships of the *managed object* and the response is a list of EntityReference elements. The value of this attribute is the relation that the *managed object* has with all of the returned entities. For example:

```
<wsdl:operation name="GetMembers"
    wsmf:relation=".../ManagerOfCollection" ...>
  ...
</wsdl:operation>
```

6.4 managementInformation WSDL Attribute

In order to aid a *manager* in the discovery of the management capabilities of a *managed object* , the WSDL attribute "wsmf:managementInformation" has been created to mark WSDL "import" elements that may point to documents containing additional management information. A *manager* SHOULD follow these references when searching for additional management capabilities. The document which is "import"ed MAY contain management information and is NOT REQUIRED to do so. This attribute helps a *manager* optimize its search for a complete definition of the management capabilities of a *managed object* . With no indication of which import statements may be useful, a *manager* MAY still perform an exhaustive search through all imported documents. The value of this attribute is "true" if the document being imported MAY contain management information.

```
<wsdl:import namespace="..."
    schemaLocation="..."
    wsmf:managementInformation="true" />
```

7. Transport Considerations

Reliability is a key feature for management systems. Today Web services are widely relying on HTTP, a non-reliable point-to-point protocol. However, Web services may use other transport protocols to provide reliability, or one of the SOAP-based reliability mechanisms such as WS-ReliableMessaging may be used. To address reliability needs, the appropriate protocol should be chosen. Nevertheless, all WSMF-Foundation implementations SHOULD support the SOAP over HTTP binding specified in the WSDL document associated with this specification. If a *managed object* supports a SOAP over HTTP binding, it MUST support the binding specified in the WSDL document associated with this specification.

8. Security Considerations

Because messages can be modified or forged, it is strongly RECOMMENDED that WSMF-Foundation implementations use WS-Security to ensure messages have not been modified or forged while in transit or while residing at destinations. Similarly, invalid or expired messages could be re-used or message headers not specifically associated with the specific message could be referenced. Consequently, when using WS-Security, signatures MUST include the semantically significant headers and the message body (as well as any other relevant data) so that they cannot be independently separated and re-used.

Messaging protocols used to communicate between a *manager* and a *managed object* are subject to various forms of replay attacks. In addition to the mechanisms listed above, messages SHOULD include a message timestamp (as described in WS-Security) within the signature. Recipients can use the timestamp information to cache the most recent messages and detect duplicate transmissions and prevent potential replay attacks.

It should also be noted that Web services are subject to various forms of denial-of-service attacks. Implementers of management systems compliant with this specification should take this into account.

In many cases the data exchanged between a *manager* and a *managed object* is sensitive and should remain private. It may be appropriate to encrypt management request and response messages to prevent the details of a resource from being exposed to unauthorized monitoring and/or control. WSMF-Foundation implementations SHOULD encrypt (as described in WS-Security) management messages when they contain sensitive data.

Any of the operations specified in this document may result in security fault messages for authentication, authorization, or other security failures. A *managed object* implementation supporting secure messages should be prepared to generate these faults, and a *manager* implementation should be prepared to receive security faults in response to messages it sends to *managed object* s.

Since a *managed object* may be managed by several *manager* s, it may be necessary to provide some type of access control to allow more trusted *manager* s a higher degree of access than lesser trusted *manager* s. WSMF-Foundation has divided attributes and operations into different *management interface* s in order to facilitate access control at a fairly course level - the interface. For every interface, the authenticated user, or the authenticated user mapped to a role, can be authorized against those managers or roles allowed to use the interface. Interfaces having different authorization needs could even use different management endpoints to make the authorization checks even easier. User authentication and the mapping of an authenticated user to a role is not defined in this specification.

In some cases, management may need to be done through a firewall. There are security implications in this type of management. A system administrator should make sure that a *manager* accessing a *managed object* from outside the firewall does not have access to information that is private to the enterprise. Some management information may be private to the enterprise. It is best to prohibit all access from outside the firewall initially, and only add capabilities for *manager* s as the need is discovered. The type of authentication used for messages from outside the firewall may need to be stronger than for messages from inside the firewall to assure the messages really arrived from the authenticated user.

9. Future Directions (Non-Normative)

There are several areas which WSMF-Foundation may address in the future. Some of these include the

following:

- WSMF-Foundation does not currently specify any timeouts or provide for controlling the timeouts for requests. As more experience is gained with implementations, the appropriate behaviors on both the *manager* and *managed object* may be specified.
- WSMF-Foundation provides a mechanism to manage many resources. Some operations, particularly in the `ManagedObjectCollection` *management interface* , may require very large arguments or responses for large numbers of *managed object* s. Future versions of WSMF-Foundation may enhance the scalability of these operations.
- The overall status of a *managed object* (healthy, failed) is useful to a *manager* . Different *manager* s have different ideas about what healthy means. Future versions of WSMF-Foundation could provide a rule-based method of determining the status of a *managed object* . The *managed object* would need to keep a different rule for each *manager* wishing to obtain the status. Notifications would be fired when the rule determines a change in status.
- WSMF-Foundation has a fixed set of relations and applications of WSMF-Foundation can add new relations. However, this is a very static method of specifying associations. Future versions of WSMF-Foundation may specify a way to dynamically add new relations as a *managed object* evolves. A notification would also be available to indicate a new or removed relation.
- WSMF-Foundation has a fixed set of attributes. Future versions of WSMF-Foundation may add a mechanism to add and remove attributes dynamically. It is also necessary to discover which attributes are currently available, and to be notified when an attribute is added or removed.
- WSMF-Foundation currently provides a limited set of meta-information about an attribute, operation, or notification. Future versions of WSMF-Foundation may include a more powerful method of describing the meta-information associated with attributes, operations, and notifications, and also to allow the dynamic modification of this meta-information.
- WSMF-Foundation alludes to *managed object* addressing security by providing an access list for each *management interface* . However, it is never specified as to how an implementation should provide this or how it is communicated to *manager* s. Future versions of WSMF-Foundation may provide the means to advertise the policy of the *managed object* . This policy would address the capability to advertise the available *management interface* s to a *manager* , in addition to advertising various other policies of the *managed object* . Of course, it will still be up to the implementation to make sure the policy is enforced.
- WSMF-Foundation specifies a Performance category in which attributes may be defined as a simple measurement. However, this falls short of what some *manager* s would like. A history of an attribute associated with performance can be very useful in establishing a trend and potentially predicting a future failure. This behavior is sometimes called a measurement. Measurements can be complex calculations based upon existing attributes. It may be necessary for different *manager* s to specify their own measurements. It is not desireable for a notification to be fired for each new measurement calculation, but rather only upon exceeding a threshold, or every N seconds, or a certain number of times. Future versions of WSMF-Foundation may provide a measurement capability.
- WSMF-Foundation provides a fixed view of a lifecycle of a resource even though it is possible for applications of WSMF-Foundation to add new substates. Future versions of WSMF-Foundation may provide a means for a *managed object* to express its lifecycle and its various states dynamically. This may include adding an operation to allow a *manager* to test the resource and/or the *managed object* .
- Future versions of WSMF-Foundation may provide more powerful methods of discovering new *managed object* s.
- WSMF-Foundation offers limited ways to manage the management infrastructure. Future versions of WSMF-Foundation may provide a more extensive means to make sure all of the management capability is running as smoothly as the business capability being managed. This may be done as an interface for managing the *managed object* rather than the resource.

10. References

10.1 Normative References

[IETF RFC 2119]
Key words for use in RFCs to Indicate Requirement Levels , S. Bradner, Author. Internet Engineering Task Force, June 1999. Available at http://www.ietf.org/rfc/rfc2119.txt. (See http://www.ietf.org/rfc/rfc2119.txt.)

[IETF RFC 2396]
Uniform Resource Identifiers (URI): Generic Syntax , T. Berners-Lee, R. Fielding, L. Masinter, Authors. Internet Engineering Task Force, August 1998. Available at http://www.ietf.org/rfc/rfc2396.txt. (See http://www.ietf.org/rfc/rfc2396.txt.)

[XML 1.0]
Extensible Markup Language (XML) 1.0 (Second Edition) , T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler, Editors. World Wide Web Consortium, 10 February 1998, revised 6 October 2000. This version of the XML 1.0 Recommendation is http://www.w3.org/TR/2000/REC-xml-20001006. The latest version of XML 1.0 is available at http://www.w3.org/TR/REC-xml. (See http://www.w3.org/TR/2000/REC-xml-20001006.)

[XML Namespace]
Namespaces in XML , T. Bray, D. Hollander, and A. Layman, Editors. World Wide Web Consortium, 14 January 1999. This version of the XML Information Set Recommendation is http://www.w3.org/TR/1999/REC-xml-names-19990114. The latest version of Namespaces in XML is available at http://www.w3.org/TR/REC-xml-names. (See http://www.w3.org/TR/1999/REC-xml-names-19990114.)

[XML Schema: Structures]
XML Schema Part 1: Structures , H. Thompson, D. Beech, M. Maloney, and N. Mendelsohn, Editors. World Wide Web Consortium, 2 May 2001. This version of the XML Schema Part 1 Recommendation is http://www.w3.org/TR/2001/REC-xmlschema-1-20010502. The latest version of XML Schema Part 1 is available at http://www.w3.org/TR/xmlschema-1. (See http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/.)

[XML Schema: Datatypes]
XML Schema Part 2: Datatypes , P. Byron and A. Malhotra, Editors. World Wide Web Consortium, 2 May 2001. This version of the XML Schema Part 2 Recommendation is http://www.w3.org/TR/2001/REC-xmlschema-2-20010502. The latest version of XML Schema Part 2 is available at http://www.w3.org/TR/xmlschema-2. (See http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.)

10.2 Informative References

[IETF RFC 2616]
Hypertext Transfer Protocol -- HTTP/1.1 , R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, Authors. Internet Engineering Task Force, June 1999. Available at http://www.ietf.org/rfc/rfc2616.txt. (See http://www.ietf.org/rfc/rfc2616.txt.)

[SOAP 1.1]
Simple Object Access Protocol (SOAP) 1.1 , D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Frystyk Nielsen, S. Thatte, D. Winer, Editors. World Wide Web Consortium, 8 May 2000. This version of the Simple Object Access Protocol 1.1 Note is http://www.w3.org/TR/2000/NOTE-SOAP-20000508. The latest version of Simple Object Access Protocol 1.1 is available at http://www.w3.org/TR/SOAP. (See http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.)

[SOAP 1.2]
SOAP Version 1.2 Part 1: Messaging Framework , M. Gudgin, M. Hadley, N. Mendelsohn, J-J. Moreau, H. Frystyk Nielsen, Editors. World Wide Web Consortium, 19 December 2002. This version of the SOAP Version 1.2 Part 1 Specification is http://www.w3.org/TR/2002/CR-soap12-part1-20021219/. The latest version of SOAP Version 1.2 Part 1 is available at http://www.w3.org/TR/soap12-part1/. (See http://www.w3.org/TR/2002/CR-soap12-part1-20021219/.)

[XML Linking]
XML Linking Language (XLink) Version 1.0 , S. DeRose, E. Maler, D. Orchard, Editors. World Wide Web Consortium, 27 June 2001. This version of the XML Linking Language 1.0

Recommendation is http://www.w3.org/TR/2001/REC-xlink-20010627. The latest version of XML Linking Language 1.0 is available at http://www.w3.org/TR/xlink. (See http://www.w3.org/TR/2001/REC-xlink-20010627/.)

[XPath]

XML Path Language (XPath) Version 1.0 , J. Clark, S. DeRose, Editors. World Wide Web Consortium, 16 November 1999. This version of the XML Path Language 1.0 Recommendation is http://www.w3.org/TR/1999/REC-xpath-19991116. The latest version of XML Path Language 1.0 is available at http://www.w3.org/TR/xpath. (See http://www.w3.org/TR/1999/REC-xpath-19991116.)

[WSDL]

Web Services Description Language (WSDL) 1.1 , E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, Authors. World Wide Web Consortium, 15 March 2002. This version of the Web Services Description Language Note is http://www.w3.org/TR/2001/NOTE-wsdl-20010315. The latest version of Web Services Description Language is available at http://www.w3.org/TR/wsdl. (See http://www.w3.org/TR/2001/NOTE-wsdl-20010315.)

[WSDL 1.2]

Web Services Description Language (WSDL) Version 1.2 , R, Chinnici, M. Gudgin, J. Moreau, and S. Weerawarana, Editors. World Wide Web Consortium. The latest version of Web Services Description Language, Version 1.2 is available at http://www.w3.org/TR/wsdl12. (See http://www.w3.org/TR/2003/WD-wsdl12-20030303.)

[WS-Events]

Web Services Events , N. Catania, Editor. Web Services Management Framework, Hewlett-Packard. (See http://devresource.hp.com/drc/specifications/wsmf/WS-Events.html.)

[WSMF-WSM]

WSMF: Web Services Model , W. Vambenepe, Editor. Web Services Management Framework, Hewlett-Packard. (See http://devresource.hp.com/drc/specifications/wsmf/WSMF-WSM.html.)