

# The WS-Management Catalog

June 2005

## Authors

Akhil Arora, Sun Microsystems, Inc.  
Josh Cohen, Microsoft Corporation  
Jim Davis, WBEM Solutions, Inc.  
Mike Dutch, Symantec Corporation  
Eugene Golovinsky, BMC Software, Inc.  
Yasuhiro Hagiwara, NEC Corporation  
Jackson He, Intel Corporation  
David Hines, Intel Corporation  
Reiji Inohara, NEC Corporation  
Christane Kämpfe, Fujitsu-Siemens Computers  
Raymond McCollum, Microsoft Corporation (Editor)  
Milan Milenkovic, Intel Corporation  
Paul Montgomery, Advanced Micro Devices, Inc.  
Alexander Nosov, Microsoft Corporation  
Abhay Padlia, Novell, Inc.  
Roger Reich, Symantec Corporation  
Larry Russon, Novell, Inc.  
Jeffrey Schlimmer, Microsoft Corporation  
Enoch Suen, Dell Inc.  
Vijay Tewari, Intel Corporation  
Kirk Wilson, Computer Associates

## Copyright Notice

(c) 2004, 2005 [Advanced Micro Devices, Inc.](#), [BMC Software, Inc.](#), [Computer Associates, Dell, Inc.](#), [Fujitsu-Siemens Computers](#), [Intel Corporation](#), [Microsoft Corporation](#), [NEC Corporation](#), [Novell Inc.](#), [Sun Microsystems, Inc.](#), [Symantec Corporation](#), and [WBEM Solutions, Inc.](#) All rights reserved.

Permission to copy and display WS-Management, which includes its associated WSDL and Schema files and any other associated metadata (the "Specification"), in any medium without fee or royalty is hereby granted, provided that you include the following on ALL copies of the Specification that you make:

- (1) A link or URL to the Specification at one of the Co-Developers' websites.
- (2) The copyright notice as shown in the Specification.

Microsoft, Intel, AMD, BMC, Computer Associates, Dell, Fujitsu-Siemens, NEC, Novell, Sun, Symantec, and WBEM Solutions (collectively, the "Co-Developers") each

agree upon request to grant you a license, provided you agree to be bound by such license, under royalty-free and otherwise reasonable, non-discriminatory terms and conditions to their respective patent claims that would necessarily be infringed by an implementation of the Specification and solely to the extent necessary to comply with the Specification.

THE SPECIFICATION IS PROVIDED "AS IS," AND THE CO-DEVELOPERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE SPECIFICATION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

THE CO-DEVELOPERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE SPECIFICATIONS.

The name and trademarks of the Co-Developers may NOT be used in any manner, including advertising or publicity pertaining to the Specifications or their contents without specific, written prior permission. Title to copyright in the Specifications will at all times remain with the Co-Developers.

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

## **Abstract**

This document describes the default metadata formats used for the WS-Management Protocol. The entire set of metadata documents available from a WS-Management service is called the "catalog".

## **Status of this document**

This corresponds to the WS-Management version of June 2005.

## **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 [RFC2119 \[1\]](#).

The above keywords are used in UPPERCASE in conformance requirements statements specifically called out in the specification using the following format:

**CR 1.3.a** - *Language for a specific conformance requirement*

...in which the chapter section forms the first part of the rule and a lowercase letter forms the actual rule identity. Use of the above keywords outside of a conformance rule is coincidental.

## **Table of Contents**

1	Introduction.....	4
1.1	Use Cases.....	5
1.2	Terminology.....	6
1.3	XML Namespaces.....	7
1.4	Catalog Organization.....	7
1.4.1	General.....	7
1.4.2	Role of the Catalog.....	8
1.4.3	Grouping of catalog entries.....	10
1.4.4	Multiple Catalogs.....	14
1.4.5	WSDL and XML Schema Content.....	14
2	<i>Resource</i> Format.....	15
2.1	Introduction.....	15
2.2	Overall Structure.....	16
2.3	<i>Resource</i> .....	18
2.4	<i>ResourceURI</i> .....	19
2.5	<i>Versioning</i> .....	20
2.6	<i>Notes</i> .....	21

2.7	<i>Vendor</i> .....	21
2.8	<i>DisplayName</i> .....	22
2.9	<i>Keywords</i> .....	22
2.10	<i>Access</i> .....	25
2.11	<i>Compliance</i> .....	25
2.12	<i>Operation</i> .....	25
2.12.1	<i>Action</i> .....	26
2.12.2	<i>SelectorSetRef</i> .....	28
2.12.3	<i>OptionSetRef</i> .....	29
2.12.4	<i>SchemaRef</i> .....	29
2.12.5	<i>FilterDialect</i> .....	30
2.12.6	<i>DeliveryModes</i> .....	31
2.13	<i>SelectorSet</i> .....	32
2.14	<i>Options</i> .....	34
2.15	<i>Relationships</i> .....	35
2.15.1	<i>Unary Relationships</i> .....	35
2.15.2	<i>Binary and N-Ary Relationships</i> .....	37
2.16	<i>Open Content</i> .....	38
3	<i>Events</i> .....	39
4	<i>WSDL Usage</i> .....	41
4.1	<i>General</i> .....	41
4.2	<i>Primary Issues</i> .....	41
4.3	<i>wsa:Action URIs</i> .....	42
4.4	<i>WS-Management Endpoint Reference Elements</i> .....	42
5	<i>Retrieving Metadata</i> .....	43
5.1	<i>Introduction</i> .....	43
5.2	<i>Subsequent Catalog Versions and Schemas</i> .....	46
5.3	<i>Catalog Location and Indirection</i> .....	47
5.4	<i>Localized Metadata</i> .....	48
5.5	<i>Retrieving XML Schema and WSDL</i> .....	50
6	<i>Appendix B : Resource XSD</i> .....	50
7	<i>Appendix C : Example Catalog Entries</i> .....	54
7.1	<i>Minimal</i> .....	54
7.2	<i>More Complex Example</i> .....	55
7.3	<i>Event and Event Source Example</i> .....	56
8	<i>References</i> .....	58

---

# 1 Introduction

**WS-Management** is a general-purpose SOAP-based systems management protocol and is based on a small number of fixed operations typical to management tasks.

For details on the protocol itself, see the associated **Web Services for Management (WS-Management)** specification, of **June 2005** [1], which is the normative definition and relates directly to the definitions in this specification.

This specification defines the default metadata formats for the discovery part of the protocol. Discovery in this context refers to discovering available resources at a particular network address or management node.

WS-Management supports the concept of multiple logical endpoints residing at the same network address, so a technique is required for discovering and understanding what management functionality those endpoints represent.

This list of available logical endpoints or "resources", their summary forms, compatible actions, schemas, and WSDL representations loosely constitute the **WS-Management Catalog**.

While WS-Management itself can work with more than one metadata format, this specification is offered as a practical starting point for organizing the management data needed by users of the protocol.

---

## 1.1 Use Cases

**WS-Management** [1] defines protocol operations for carrying out management operations: getting and setting values, enumerating tables, event subscriptions, and execution of custom methods.

However, the protocol operations require that the sender properly form an address to the management "endpoint". A management endpoint is a piece of code or instrumentation which can service the request. Discovering the availability and names of such endpoints is a critical step in the work flow.

For example, to retrieve the amount of remaining free space on a hard drive, the WS-Transfer "Get" operation is used. Within that SOAP message, there are elements which target that message at the handler or provider which can retrieve that information.

The total combination of those addressing elements is referred to as the **endpoint reference (EPR)**, defined in WS-Addressing [2] and is further clarified in the WS-Management specification. All WS-Management protocol operations must have a valid EPR so that the operation can be targeted at the right handler. The EPR for the example would consist of a URI indicating the "disk" object, and a selector indicating which "drive" is required (the drive letter).

The WS-Management catalog consists of metadata entries for each resource which allow users and tools to discover how to construct valid EPRs to access these resources.

Typical use cases are:

- An interactive user using command line tools wants to get or set a value on a remote server. To properly target the operation, the user must know something about the naming of the resource.
- A developer wants to use a code generation tool to produce strongly-typed SOAP proxies to access specific resources. To do this, WSDL definitions for the resources are required.
- A command-line or GUI tool needs to be able to list the entire catalog or subsets of it, so that authors of management policies can determine what resources are available on a specific node.
- Policy engines want to ensure specific versions of resources are installed on the target nodes prior to deploying monitoring software which relies on those resources
- Software which can access arbitrary SOAP methods defined in WSDL needs a way to get the WSDL document so that custom methods for a resource can be reliably accessed. A command-line tool which is WSDL-aware can expose a simple command-line interface to management-related methods in this way.
- Engines which forward events need to know what events are available and where to send the subscription to receive those events

---

## 1.2 Terminology

The following WS-Management-specific terms are used in this document:

### **Client**

The client application using the Web services defined in this document to access the management service.

### **Service**

An application that provides management services to clients by exposing the web services defined in this document. A service typically is equivalent to the network "listener" and is associated with a physical transport address and is essentially a type of manageability access point.

### **Resource**

An endpoint which represents a distinct type of management operation or value. A service exposes one or more resources and some resources can have more than one instance. In this sense, a resource is similar to a "class" or a database table, and an instance is similar to an instance of the class or a row in the table.

### **Selector**

A resource-relative identifier and value which is an instance-level discriminant. This is essentially a filter which isolates the instance.

### **Catalog**

The entire set of available resource metadata documents which together describe all the resources that a particular service exposes. There is one catalog per service.

---

## 1.3 XML Namespaces

The following XML namespaces are used within this document and the WS-Management protocol itself. The choice of the namespace prefix is arbitrary and not semantically significant.

### Prefixes and XML namespaces used in this specification.

Prefix	XML Namespace	Specification(s)
wsmancat	<a href="http://schemas.xmlsoap.org/ws/2005/06/wsmancat">http://schemas.xmlsoap.org/ws/2005/06/wsmancat</a>	This specification
wsmam	<a href="http://schemas.xmlsoap.org/ws/2005/06/management">http://schemas.xmlsoap.org/ws/2005/06/management</a>	WS-Management Protocol
s	<a href="http://www.w3.org/2003/05/soap-envelope">http://www.w3.org/2003/05/soap-envelope</a>	SOAP 1.2 [ <a href="#">SOAP 1.2</a> ]
xs	<a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a>	XML Schema [ <a href="#">Part 1</a> , <a href="#">2</a> ]
wSDL	<a href="http://schemas.xmlsoap.org/wSDL">http://schemas.xmlsoap.org/wSDL</a>	WSDL/1.1 [ <a href="#">WSDL 1.1</a> ]
wsa	<a href="http://schemas.xmlsoap.org/ws/2004/08/addressing">http://schemas.xmlsoap.org/ws/2004/08/addressing</a>	WS-Addressing [ <a href="#">WS-Addressing</a> ]
wse	<a href="http://schemas.xmlsoap.org/ws/2004/08/eventing">http://schemas.xmlsoap.org/ws/2004/08/eventing</a>	WS-Eventing [ <a href="#">WS-Eventing</a> ]
wsen	<a href="http://schemas.xmlsoap.org/ws/2004/09/enumeration">http://schemas.xmlsoap.org/ws/2004/09/enumeration</a>	WS-Enumeration [ <a href="#">WS-Enumeration</a> ]
wxf	<a href="http://schemas.xmlsoap.org/ws/2004/09/transfer">http://schemas.xmlsoap.org/ws/2004/09/transfer</a>	WS-Transfer [ <a href="#">WS-Transfer</a> ]

---

## 1.4 Catalog Organization

### 1.4.1 General

The catalog is a simple flat list of resource metadata documents. Each entry provides

- (1) A unique identifier for a real-world resource type (the ResourceURI).
- (2) Descriptive information so that browsers and other tools can publish available resource names to users and tool authors
- (3) Keyword and other search information to allow users to do searches over large catalogs
- (4) Relationships between catalog entries and the outside world, including external references to help or other documentation on the Web or other locations.
- (5) References to the protocol Action level (access profile) of the resource
- (6) References to the appropriate WSDL definitions and XML Schemas for the specific catalog entry

The intent is to allow services to publish information **about available resources** and have it discovered and processed automatically by a wide variety of tools, and to allow more or less automatic WS-Management bindings to be built.

Since most of the information in the catalog is closely related to WS-Management itself, the catalog can often be used incidentally at runtime by the stack itself to analyze, verify, and route incoming requests.

The catalog is a list of resource *types*, not instances. A simple catalog entry fragment follows. The fields will not be discussed in detail in this particular section, but a picture of a catalog entry is useful in the following discussions:

```
(1) <Resource
(2)   xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat "
(3)   xmlns:xs="http://www.w3.org/2001/XMLSchema "
(4)   lang="en-us" >
(5)
(6)   <ResourceURI> http://schemas.acme.com/2005/06/samples/physicalDisk </ResourceURI>
(7)
(8)   . . . .
(9)
(10) </Resource>
```

Each catalog entry is called a **Resource** (line 1) for each object type that can be accessed via the protocol. The **ResourceURI** (line 6) is the same ResourceURI that would be used in the WS-Management protocol operations at the instance level, (such as wxf:Get, or wsen:Enumerate) to actually get the instrumented values described by the catalog entry.

Note that the resource definition is at the *type* level. That is, only *one* resource definition will exist in the catalog for all instances of the resource. In the example above, there is one definition for the "disk" object, regardless of how many disks might be present on the system, and selectors are used to specify which instance is required when the instance-level protocol operations are used. Conversely, note that there is *not* one catalog entry for each *instance* of a resource type.

The rest of the catalog entry contains details about how to access the resource, its capabilities, and where to find more information.

A catalog entry may represent an "object" or complex XML Infoset which models a real-world object, or may represent an individual property about a real-world object. This specification places no constraints on the granularity of a catalog entry. A single catalog entry may be used, or a set of catalog entries may together expose the real-world object.

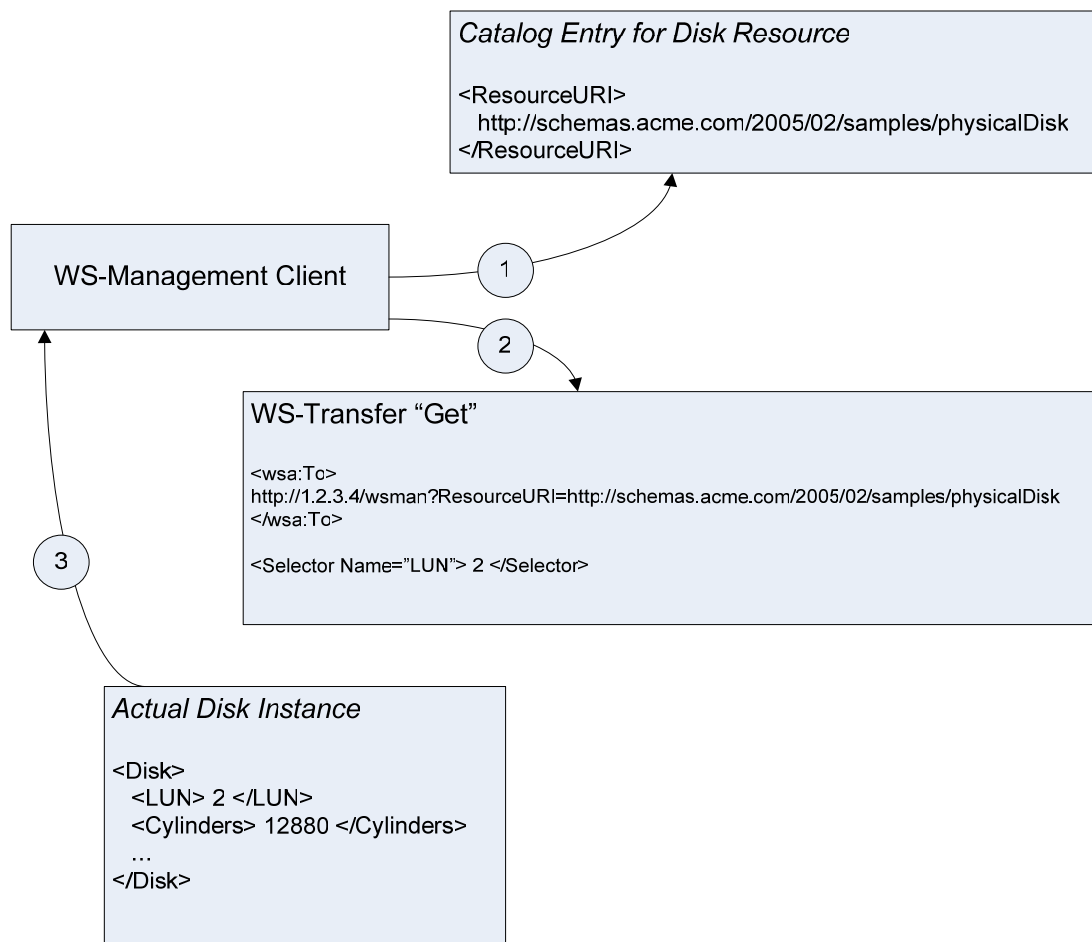
It is typical that catalog entries are at an "object" or "class" level, meaning that the catalog entry represents a "class" of entry and individual WS-Management protocol operations can return or manipulate "instances".

### **1.4.2 Role of the Catalog**

Each Resource document is a separate catalog entry. The ResourceURI within that document is the identity of the catalog entry, and also the identity of the real-world representation object to be accessed by a WS-Management protocol operation at the instance level.

When the ResourceURI is used in a wxf:Get at the protocol level, the real-world resource is retrieved. The catalog entry describes the entity, what its capabilities are and what will be returned if it is accessed. Retrieving the catalog entry itself is also done using wxf:Get, but with a different technique.

In the diagram below, the WS-Management client "discovers" the name and addressing technique for any given disk (step 1) by consulting the catalog. It then builds a request for a particular disk using this information and by issuing a WS-Management protocol operation at the instance level, such as a WS-Transfer "Get" (step 2). Finally, the management service returns the actual disk instance (step 3) to the client:

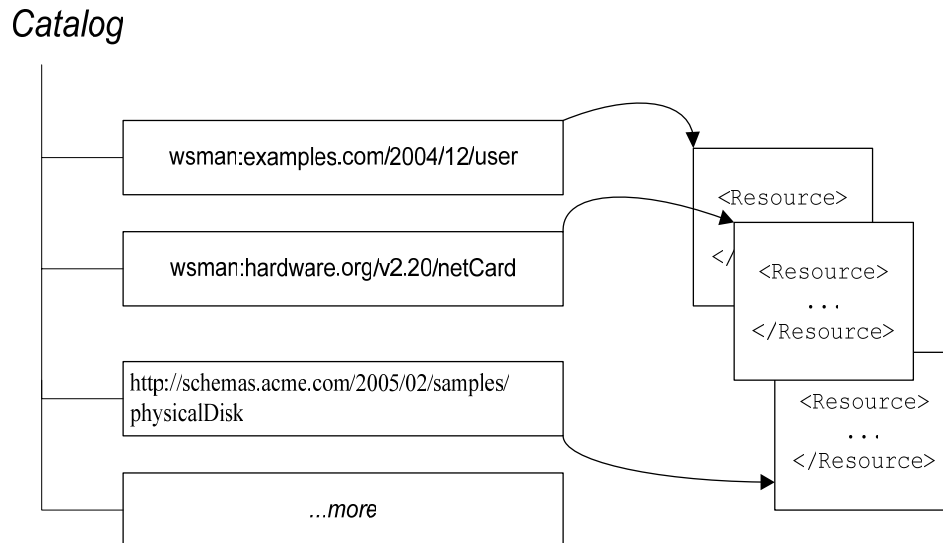


It is important to realize that the client accesses the resource catalog entry to discover how to build an address to a resource at the protocol level and how to discover what capabilities the resource has.

At runtime, the WS-Management protocol may make no use of the catalog entry at all, but may simply directly access the resource.

The catalog answers the question "Tell me about X", whereas the WS-Transfer:Get protocol operation using that same identity instructs the service to "Now, go and *get* X".

The basic organization of catalog entries is flat. All resources are described in separate XML documents, and all such documents are organized as a simple linear list, where each document is associated with the Resource URI itself, as described in the previous section:



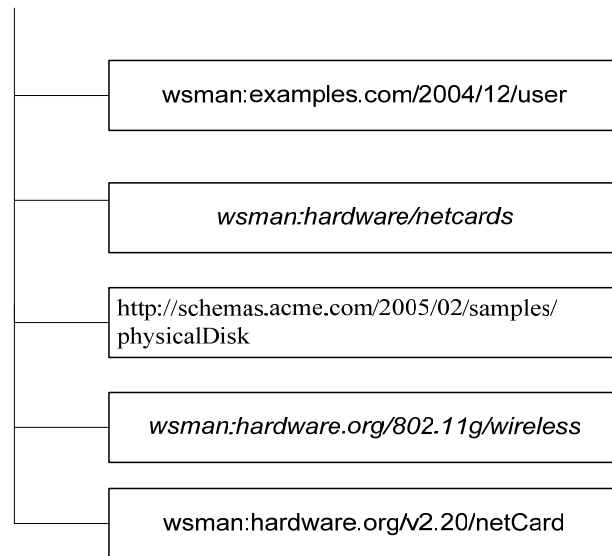
Each entry is indexed by its ResourceURI, so any reference to the ResourceURI is an implicit or explicit reference to the associated catalog entry. A catalog entry thus acts as a "class" definition where the class name is the ResourceURI. While vendors may in fact establish meaningful URI organization to model containment, this specification leaves the vendor free to establish any such usage.

### **1.4.3 Grouping of catalog entries**

All organization of the catalog entries into semantic groups is done by stitching relationships over the top of these entries in the `<Relationships>` element.

For example, if three of the hypothetical resources in the following catalog (`wsman:hardware/netcards`, `wsman:hardware.org/802.11g/wireless` and `wsman:hardware.org/v2.20/netcard`) have a hierarchical relationship:

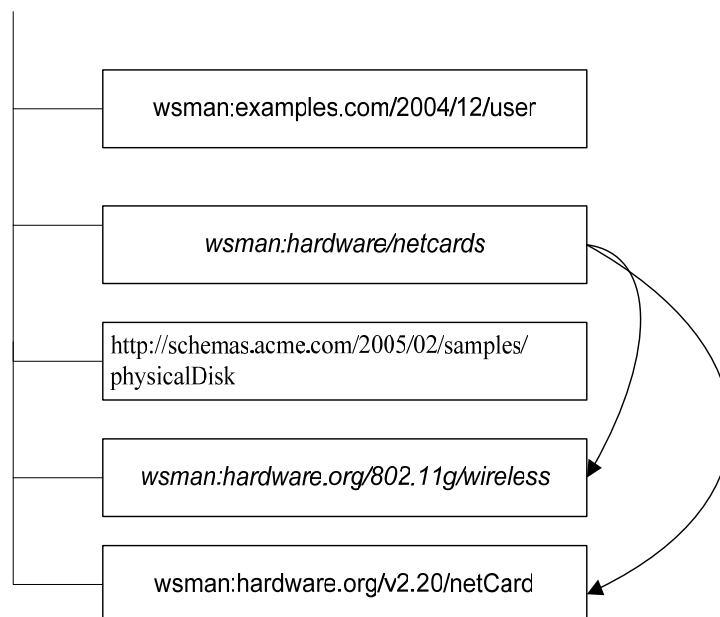
## Catalog



...then this relationship is properly expressed using metadata within the entries, as opposed to using structural constraints on the catalog or the URI format itself.

The `wsman:hardware/netcards` entry could be supplemented with references (or "pointers") to the two child entries:

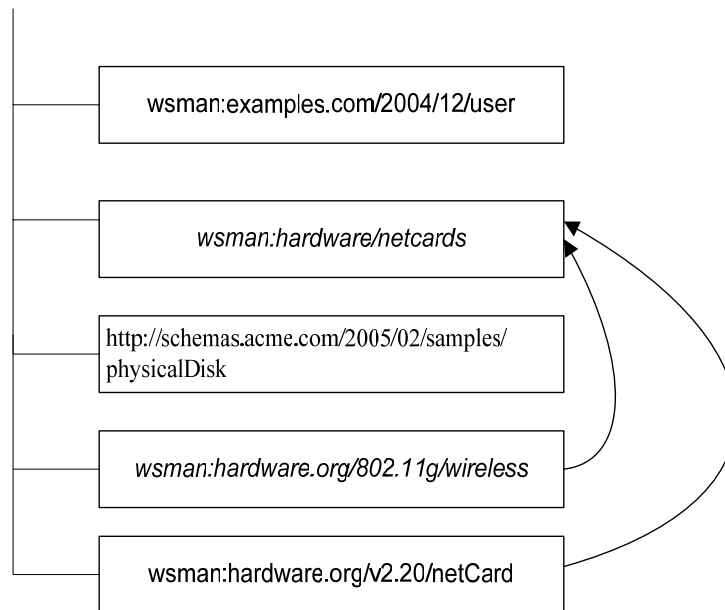
## Catalog



In this case, the parent object points to the children and models the containment.

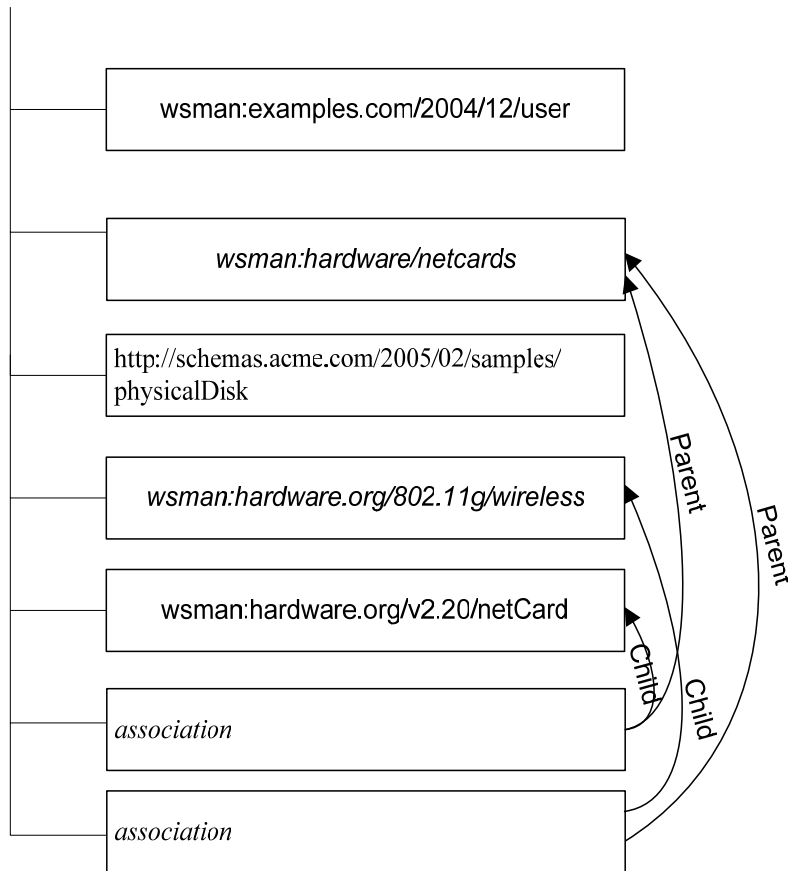
Alternately, the children might point to the parent and model the containment by asserting who the parent is

## Catalog



...or an association model might be defined in which two new catalog entries each contain two references, one to the parent and one to the child to establish the relationship between them:

## Catalog



The mechanisms for describing relationships are defined within the catalog itself. This specification predefines several relationship types, but users may choose to model new types of relationships or model well-known relationships a different way.

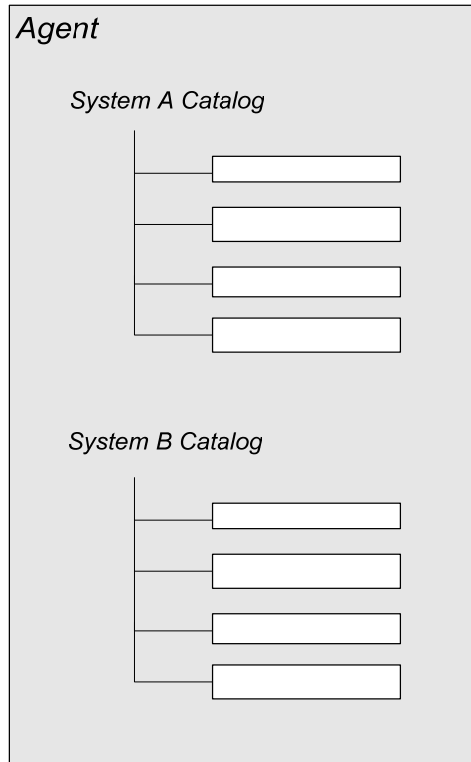
Using this flexible relationship definition model, different vendors can establish specific or private URI structuring and do not have to arrive at a common URI-based standard for describing containment or other relationships between catalog entries. Because the relationships are explicit links within the entries, the grouping can be built over random or dissimilar URI structures if required.

Note that any relationships which may exist **between instances** returned in the actual WS-Management protocol operations are not modeled in the catalog per se, but are part of the XML Schema definitions for those objects.

Of course a designer may take advantage of the freedom of being able to use any URI model whatsoever to relate a number of catalog entries. The only restriction is that the URI must contain the vendor DNS name:

### 1.4.4 Multiple Catalogs

If a WS-Management agent service supports multiple logical services, then each has its own catalog. In essence, these are nothing more than separately addressed services, but may in fact be hosted by the same implementation. They are logically independent and relationships between the systems are not modeled in the catalog or catalogs themselves:



"Subservices" within the protocol agent itself may share a catalog for efficiency reasons in terms of the implementation, but behaviorally, each is exposing its own catalog.

### 1.4.5 WSDL and XML Schema Content

The Resource format is not a replacement for WSDL or XML Schema, but rather a supplementary document which helps WS-Management discover how to find and interpret the capabilities of management resources (including WSDL definitions and XML schemas). In practice, a typical package for a real-world component would be:

- (1) One or more catalog entries
- (2) One (or more) XML schemas for the objects to be instrumented (corresponding to the catalog entries).
- (3) One or more WSDL documents which describe any custom methods and any strong schematization of standard WS-Management operations.

In general, a complete set of these XML documents will be required to fully support all types of users:

It is important that these documents retain their historical separation and factoring. SOAP toolkits need WSDL, but validating XML parsers need the XML schema and not the WSDL. Similarly, WS-Management-aware tools need the Resource description and may not need the others, and some complex tools may need all three. Further, the XML schemas and WSDL documents may be used for other purposes and should not be locked down for management purposes only.

For example, a WSDL definition for a Web Service which exposes credit card transaction capabilities may expose the primary methods relating to charging or crediting credit cards, and may also expose the methods relating to management of the service itself, such as a Reset method, or may expose statistical information such as the number of transactions per hour.

Neither WSDL nor XML schema documents alone can fully describe all of the information typically required by management clients, which is why the <Resource> format exists as a new WS-Management metadata type. For example, XML Schema is not designed to describe which protocol operations apply to a given schema element, and WSDL is not designed to describe keywords or tags about a resource to promote intelligent searching.

Because WSDL and XML schema documents can require immense amounts of space and have very specific uses, they do not constitute the body of the catalog itself and often reside in different locations than the primary catalog that the service may expose directly. In many implementations, the Resource entry will be resident with the service, while the WSDL and XML schema documents will be publicly available via http-based Web addresses. However, if the service is resource-constrained, even the catalog itself may reside on the Web.

The structure of the Resource entry, and the techniques for locating and enumerating the Resource entries, WSDL, and XML schemas are covered in subsequent sections.

---

---

## **2 Resource Format**

---

### **2.1 Introduction**

The Resource Format includes descriptive information about resources and their relationships. It is intended for WS-Management aware tools which need to browse and discover available resources.

The metadata formats for the catalog and the techniques for discovering them are decoupled. That is, if additional formats are required in the future, the newer formats

can coexist with older ones. The process for discovering and enumerating them remains the same, however.

---

## 2.2 Overall Structure

Each resource that is accessible through WS-Management has a summary metadata format that describes the resource. The catalog consists of a simple linear list, potentially very large, of catalog entries consisting of <Resource> documents. Here is an example of a catalog entry with a number of capabilities:

```
(1) <Resource
(2)   xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat"
(3)   xmlns:wsmancat="http://schemas.xmlsoap.org/ws/2005/06/wsmancat"
(4)   xmlns:xs="http://www.w3.org/2001/XMLSchema"
(5)   lang="en-us">
(6)
(7)   <ResourceURI> http://schemas.acme.com/2005/06/samples/physicalDisk </ResourceURI>
(8)   <Notes> Describes access to a physical disk unit </Notes>
(9)   <Vendor> Acme Corporation </Vendor>
(10)  <DisplayName> ACME Super Disk </DisplayName>
(11)
(12)  <Keywords xmlns:acme="http://keywords.acme.com/2005/06/hardware">
(13)    <Keyword>acme:Disk</Keyword>
(14)    <Keyword>acme:Storage</Keyword>
(15)  </Keywords>
(16)
(17)  <Access>
(18)    <Compliance> http://schemas.xmlsoap.org/ws/2005/06/management </Compliance>
(19)    <Operation>
(20)      <Action>
(21)        http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
(22)      </Action>
(23)      <SelectorSetRef Name="s1"/>
(24)      <SchemaRef xmlns:ac="schemas.acme.com/2005/06/hardware">
(25)        ac:PhysDisk
(26)      </SchemaRef>
(27)    </Operation>
(28)
(29)    <Operation>
(30)      <Action>
(31)        http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate
(32)      </Action>
(33)      <SchemaRef xmlns:ac="schemas.acme.com/2005/06/hardware">
(34)        ac:PhysDisk
(35)      </SchemaRef>
(36)      <OptionSetRef Name="os1"/>
(37)    </Operation>
```

```

(38)
(39)     <Operation
(40)         WsdPort="DiskOps"
(41)         WsdRef="http://schemas.acme.com/2005/06/hardware/wsdl">
(42)         <Action> http://schemas.acme.com/2005/06/hardware/disk/format </Action>
(43)         <Action> http://schemas.acme.com/2005/06/hardware/disk/powerOff </Action>
(44)         <Action> http://schemas.acme.com/2005/06/hardware/disk/powerOn </Action>
(45)     </Operation>
(46)
(47)     <SelectorSet Name="s1">
(48)         <Selector Name="LUN" Type="xs:unsignedInt">
(49)             The logical unit number of the disk
(50)         </Selector>
(51)     </SelectorSet>
(52)
(53)     <OptionSet Name="os1">
(54)         <Option Name="Verbosity" Type="xs:unsignedInt">
(55)             Controls verbosity of output 0 (least) to 10 (most)
(56)         </Option>
(57)     </OptionSet>
(58) </Access>
(59)
(60) <Relationships>
(61)     <Relationship Type="wsmancat:VendorURL" Role="wsmancat:External"
(62)         Ref="http://www.acme.com"/>
(63) </Relationships>
(64)
(65) </Resource>

```

The **ResourceURI** element (line 7) contains the URI that defines the resource type that this catalog entry refers to. This is used in the `wsa:To` block of the SOAP message to target the operation to this resource.

The **Notes** section (line 8) contains any descriptive text in the language specified in the `xml:lang` attribute on line 4.

The **Vendor** (line 9) indicates which vendor published this resource.

The **DisplayName** element (line 10) provides a short caption to describe this resource, for use in tools and utilities. It may be localized into the language specified in the `lang` attribute.

The **Keywords** section (lines 12-15) provides a list of QName-based keywords which allow search tools to find catalog entries with specific keywords. As many keywords as appropriate can be added.

The **Access** block (17-58) provides detail on how the resource can be accessed, its selectors, access options, protocol Action level, and pointers to any WSDL and XML

schema definitions. If the catalog entry supports more than one type of access, more than one Access block can be defined.

The **Compliance** section (line 18) references the version of the WS-Management protocol (or other management protocol) that the resource was designed to operate with. This may be repeated if the resource supports more than one version of the protocol.

There are one or more **Operation** elements (lines 19, 29, 39) which indicate which operations the resource is compatible with. The first two examples indicate that the resource was designed to work with wxf:Get (WS-Transfer::Get) and wsen:Enumerate (wsen:Enumerate). Note that only the wxf:Get requires Selectors, and the reference to the correct SelectorSet set is made. The example on line 39 describes that three resource-specific methods are available (lines 42-44) and described in the specified WSDL document and port. Note that WSDL-based methods may use or omit the SelectorSetRef, depending on whether Selectors are used to access methods for specific instances or whether the method applies globally to all instances of the resource.

Each **SelectorSet** has a Name (line 47) for use in cross-referencing operations with the required selectors. Within an Operation, each **SelectorSetRef** (line 23) defines any selectors required to access this resource instance when protocol operations are under way. Multiple sets of Selectors can be defined.

The **OptionSet** element (line 53) describes any Option values that may apply to operations. In this case, the "Enumerate" operation beginning on line 29 makes use of this OptionSet on line 36.

The **Relationships** section (lines 60-63) is where links are defined which can point to other catalog entries or external references for any purpose.

Each of these sections is covered in detail in the following chapters.

---

## 2.3 Resource

The overall catalog entry is a <Resource> block, which establishes the namespace bindings and the language of the catalog entry using RFC 3066 language codes (using the **xml:lang** attribute on line 4):

```
(1) <Resource
(2)   xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat "
(3)   xmlns:xs="http://www.w3.org/2001/XMLSchema"
(4)   xml:lang="en-us"
(5)   >
(6)
(7)   ... metadata for a single resource...
(8)
(9) </Resource>
```

**CR 2.3.a:** Each entry *MUST* describe exactly one Resource type, although each Resource type can have many capabilities and addressing modes.

**CR 2.3.b:** The **xml:lang** attribute *SHOULD NOT* be omitted if any language-specific content, such as <Notes> is present in the entry.

**CR 2.3.c:** The list of all available resource types for an agent *SHOULD* appear in the catalog for that given agent, subject to security constraints. That is, there *SHOULD* be one entry for each individual resource accessible over WS-Management. The catalog itself *MAY* be located at a different address than the agent itself (such as on a web site), but in that case the agent *MUST* provide the network or internet address of the relevant catalog.

While normally all entries should be present in the catalog, in certain cases security-related issues come into play. The user may not be authorized to see all the available endpoints, for example.

**CR 2.3.d:** A catalog entry *MAY* represent resource types down to the property-level where each catalog entry represents a single simple value, or it *MAY* represent resource types at an "object" level or a property bag. This specification places no constraints on the granularity of access.

**CR 2.3.e:** The catalog entry *MAY* represent a singleton resource or a collection of resource instances of the same type..

---

## 2.4 ResourceURI

The ResourceURI (line 7) forms the identity of the catalog entry, and also the identity of the real-world representation object to be accessed by a WS-Management protocol operation.

Example:

```
(1) <Resource>
(2)   xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat "
(3)   xmlns:xs="http://www.w3.org/2001/XMLSchema "
(4)   xml:lang="en-us "
(5)   >
(6)
(7)   <ResourceURI> http://schemas.acme.com/2005/06/samples/physicalDisk </ResourceURI>
(8)
```

The ResourceURI in the catalog becomes part of the endpoint (suffixed to the wsa:To address, as specified in the WS-Management specification).

**CR 2.4.a:** *The ResourceURI is a REQUIRED element and defines the URI used to address the resource in SOAP messages. This MUST be a valid URI according to RFC 2396.*

**CR 2.4.b** *If no URI scheme is already established for the resource, then **wsman:** MAY be used as a default prefix. The **http://** prefix is also permitted, and is preferred if the resource also has an associated Web address with the same URL which contains further documentation. Any other scheme is also permitted.*

**CR 2.4.c:** *After the scheme, the first component MUST be the vendor DNS name, followed by an arbitrary vendor-defined structural layout which conforms to RFC 2396 restrictions.*

**CR 2.4.d:** *A catalog entry MUST NOT reuse a ResourceURI with a different meaning than the one originally published, i.e., it may not represent an XML schema that differs from the one originally published for that URI. A new ResourceURI must be published along with the old one if any schematic change occurs or if any change in supported protocol operations occurs*

**CR 2.4.e:** *For any set of related resources, the catalog entries MUST NOT rely on URI structure alone to describe the relationships. For any relationships which exist between catalog entries, Relationships metadata must be used to describe these so that all tools will interoperate. A set of resources MAY additionally be related using URI token structuring itself, but this is in addition to the presence of Relationships block.*

**CR 2.4.f:** *The ResourceURI for a given resource MUST be unique within the catalog for that service.*

The resource is required to be unique within the scope of service as identified by its wsa:Address.

---

## **2.5 Versioning**

Resources should be versioned by their ResourceURI. Otherwise, clients cannot count on consistent behavior. Even if fully backward compatible changes are made to the processing code behind the Resource (the service), a change in the ResourceURI is still required, or else clients which take advantage of the new functionality will break if they access a Resource of the same name which does not have the updates.

For this reason, the ResourceURI acts both as an identity and a versioning mechanism. Updates should only be performed if they are non-functional.

Typically, a major update becomes a new catalog entry and a relationship between the old and new entries is established to designate the new one as an upgrade of the previous one.

If vendors need private versioning schemes, they may use the open-content extensions provided for the Resource format, or may use the Keywords mechanism to convey versioning information.

**CR 2.5.a:** Where applicable, the version information *SHOULD* be encoded into the ResourceURI itself.

---

## 2.6 Notes

The Notes element (line 8) is for text-based description. This is intended for a short "help" text on the nature of the resource, and should not be used as a caption or user-friendly name.

```
(1) <Resource
(2)   xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat "
(3)   xmlns:xs="http://www.w3.org/2001/XMLSchema "
(4)   xml:lang="en-us">
(5)
(6)   <ResourceURI> http://schemas.acme.com/2005/06/samples/physicalDisk </ResourceURI>
(7)   <Version> 1.0 </Version>
(8)   <Notes> Describes access to a physical disk unit </Notes>
(9)   ...
```

**CR 2.6.a:** The Notes element is *OPTIONAL*.

**CR 2.6.b:** If the **xml:lang** attribute is present (line 4), the language of Notes or other descriptive elements *MUST* be the specified language.

**CR 2.6.c:** If any descriptive element such as Notes is used, the **xml:lang** attribute *MUST* be present to identify the language in use.

---

## 2.7 Vendor

Describes the vendor who produced the resource (line 9):

```
(1) <Resource
(2)   xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat "
(3)   xmlns:xs="http://www.w3.org/2001/XMLSchema "
(4)   xml:lang="en-us"   >
(5)
(6)   <ResourceURI> http://schemas.acme.com/2005/06/samples/physicalDisk </ResourceURI>
(7)   <Version> 1.0 </Version>
(8)   <Notes> Describes access to a physical disk unit </Notes>
(9)   <Vendor> Acme Corporation </Vendor>
...

```

This is a vendor-specific string.

**CR 2.7.a:** *The Vendor element is OPTIONAL.*

**CR 2.7.b:** *The Vendor element SHOULD identify the vendor, either using a DNS name or other widely recognized name, such as the official corporate name of the vendor and/or its stock symbol. Alternately, mailing addresses, URLs, etc. may be used.*

---

## 2.8 DisplayName

The DisplayName is used to provide a simple caption for the object in text-based lists. It is distinct from Notes in that it should be short and suitable for inclusion in tables and GUI displays, whereas Notes is typically a more verbose description.

This is the user-friendly name which corresponds to the ResourceURI (line 11).

```
(1) <Resource
(2)   xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat"
(3)   xmlns:xs="http://www.w3.org/2001/XMLSchema"
(4)   xml:lang="en-us">
(5)
(6)   <ResourceURI> http://schemas.acme.com/2005/06/samples/physicalDisk </ResourceURI>
(7)   <Version> 1.0 </Version>
(8)   <Notes> Describes access to a physical disk unit </Notes>
(9)   <Vendor> Acme Corporation </Vendor>
(10)  <DisplayName> ACME Super Disk </DisplayName>
(11)  ...
```

**CR 2.8.a:** *The DisplayName element is OPTIONAL. There is no requirement that the DisplayName value be unique.*

**CR 2.8.b:** *The DisplayName element SHOULD consist of one to five words in the language specified by the **xml:lang** attribute (line 4).*

---

## 2.9 Keywords

The **Keywords** element is where standard and user-defined keywords can be attached to the entry to help search tools and engines. Since catalogs will have a tendency to become too large to be directly browsed from end to end, ways of doing useful searches should be accommodated.

This specification defines a standard technique for referencing keyword lists to help classify the catalog entry and to aid in effective searching. These lists may be based on

cross-vendor standards or may be designed by individual vendors. Since keywords are based on XML **QNames**, each keyword list is semantically separate and may be maintained by different organizations or vendors.

The binding of the XML namespace for the keyword must be in scope according to normal XML rules, although it would be typical to declare the namespace binding in the **Keywords** element itself:

```
(1) <Resource
(2)   xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat"
(3)   xmlns:xs="http://www.w3.org/2001/XMLSchema"
(4)   xml:lang="en-us"   >
(5)
(6)   <ResourceURI> http://schemas.acme.com/2005/06/samples/physicalDisk </ResourceURI>
(7)   <Version> 1.0 </Version>
(8)   <Notes> Describes access to a physical disk unit </Notes>
(9)   <Vendor> Acme Corporation </Vendor>
(10)  <DisplayName> ACME Super Disk </DisplayName>
(11)
(12)  <Keywords xmlns:acme="http://keywords.acme.com/2005/06/hardware">
(13)    <Keyword>acme:Disk</Keyword>
(14)    <Keyword>acme:Storage</Keyword>
(15)  </Keywords>
```

More than one namespace binding may be established to bring several keyword lists into scope:

```
(16)  <Keywords
(17)    xmlns:acme="http://keywords.acme.com/2005/06/hardware"
(18)    xmlns:widg="http://keywords.widgets.com/2004/10/storage/keywords">
(19)
(20)    <Keyword>acme:Disk</Keyword>
(21)    <Keyword>acme:Storage</Keyword>
(22)    <Keyword>widg:DiskDrive</Keyword>
(23)
(24)  </Keywords>
```

There is no limit to the number of keywords or XML namespace bindings that are allowed.

The actual namespaces being referenced may refer to "real" or "implied" documents which describe the meanings of the keywords. This specification does not establish a standard for describing such XML QNames for this purpose. If a vendor defines keywords, a document describing the QName usage should be placed at a publicly accessible location, typically the http:// address which forms the XML namespace URI itself.

Keywords may be used for any purpose which will assist users in searching over large catalogs.

## Examples:

The keyword may be related to whether the component is related to hardware or software, or related to specific kinds of devices:

```
<Keyword> sys:Hardware <Keyword>  
<Keyword> sys:StorageDevices </Keyword>
```

The keyword may indicate the cost of accessing the entire entry, indicating to the client that a long wait may be expected:

```
<Keyword>acme:TimeExpensive</Keyword>
```

The keyword may indicate compliance with a standards body:

```
<Keyword>ansi:J23100E-Compliant</Keyword>
```

The keyword may indicate a vendor-specific identification or versioning-scheme:

```
<Keyword>acme:version2-1A2-300-1</Keyword>
```

The keyword may indicate the nature of the component in a taxonomy:

```
<Keyword> acme:TopLevelComponent </Keyword>  
<Keyword> acme:Subcomponent </Keyword>  
<Keyword> acme:DriverInstrumentation </Keyword>  
<Keyword> acme:NetworkService </Keyword>
```

The keyword may indicate other names of the resources defined by an alternate naming scheme:

```
<Keyword> SMASH:cpu </Keyword>  
<Keyword> SMASH:disk </Keyword>  
<Keyword> SMASH:sensor </Keyword>
```

**CR 2.9.a:** *The Keywords element is OPTIONAL. If present, the value MUST be qualified by an XML namespace prefix.*

**CR 2.9.b:** *If a keyword from a standard published list applies, the keyword SHOULD be included in the list, even at the expense of large numbers of keywords being present. Over time, this will facilitate searching.*

**CR 2.9.c:** *Alternate keywords from other lists which overlap with standard keywords SHOULD NOT result in the omission of the standard keyword. Some clients may not be aware of some of the lists.*

This specification does not establish a standard for keyword definition documents.

---

## 2.10 Access

The Access block is a wrapper for several of the elements discussed in following sections. All of the entries relate to the protocol access to the resource.

```
(1) ...
(2) <Access>
(3)   <Compliance>... Documents protocol-level Action
(4)   <Operation> ... Documents available Actions
(5)   <SelectorSet>... Documents available Selectors
(6)   <OptionSet> ... Documents available Options
(7)   <FilterDialect> ... Lists any available filter dialects
(8)   <DeliveryMode> ... List an event delivery mode
(9) </Access>
```

**CR 2.10.a:** *The Access block is REQUIRED.*

---

## 2.11 Compliance

The Compliance element documents which versions of the protocol that this resource is compliant with. In general, this will be the namespace of the WS-Management protocol version supported by the resource:

```
(1) <Access>
(2)   <Compliance>
(3)     http://schemas.xmlsoap.org/ws/2005/06/management
(4)   </Compliance>
(5)   ...
```

Note that since the WS-Management protocol of the specified version implies dependencies on many underlying specifications, there is no need to reference all of them.

The Compliance element may be repeated to indicate compatibility with more than one protocol version or with unrelated protocols.

**CR 2.11.a:** *The Compliance element in the catalog entry MUST contain at least one Compliance element referencing the supported protocol version.*

---

## 2.12 Operation

The Operation element documents a valid operation supported by the current resource. This consists of the following content:

```
(1) <Access>
(2)   <Action> ... </Action>
(3)
(4)   <Operation>
(5)     <Action> wsa:Action URI for requests
(6)     <SelectorSetRef> Local ID of applicable selector set
(7)     <OptionSetRef> Local ID of available OptionSet
(8)     <SchemaRef> ...the schema QName applicable to this operation
(9)     <FilterDialect> ...for enumeration or subscription
(10)    <DeliveryMode> ...for event sources
(11)  </Operation>
(12)  ...
```

**CR 2.12.a:** The catalog *SHOULD* contain one or more <Operation> elements which describe what operations the Resource is compatible with.

## 2.12.1 Action

The Action element indicates the supported action or semantic for this resource. If more than one action applies, then it may be repeated as long as the remaining elements all apply to the additional Action URI as well.

**CR 2.12.b:** An Action element is *REQUIRED* as the first element of an Operation block.

More than one Action may be listed, as long as the remaining elements are also true for that Action RI. If the same Action URI is valid with different SelectorSets, OptionSets, etc., then a separate Operation block can be defined for it and the Action URI repeated in that block.

This Action URI is typically one or more of the well-known action URIs referenced in WS-Management:

```
(13) <Access>
(14)   <Compliance> ... </Compliance>
(15)
(16)   <Operation>
(17)     <Action>
(18)       http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
(19)     </Action>
(20)     <Action>
(21)       http://schemas.xmlsoap.org/ws/2004/09/transfer/Put
(22)     </Action>
(23)   ...
```

The above example states that both Get and Put from WS-Transfer are supported, and both are required to support identical Selectors, Options, SchemaRef, and anything else defined in the current Operation block.

If on the other hand Put supports different Options than Get, then a separate Operation block would have been required:

```
(24)    ...
(25)    <Operation>
(26)      <Action>
(27)        http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
(28)      </Action>
(29)      <OptionSet> opt1 </Options>
(30)    ...
(31)    </Operation>
(32)
(33)    <Operation>
(34)      <Action>
(35)        http://schemas.xmlsoap.org/ws/2004/09/transfer/Put
(36)      </Action>
(37)      <OptionSet> opt2 </Options>
(38)    ...
(39)    </Operation>
```

In fact, it is typical that Get and Put would indeed have different options, although they would often share selectors.

The following table indicates the expected Action URI values. Note that indicating support for a wsa:Action URI is equivalent to declaring support for any associated response messages, faults, and other required messages associated with that message.

Action URIs defined for this specification:

Action URI	Meaning
http://schemas.xmlsoap.org/ws/2004/09/transfer/Get	Supports WS-Transfer:Get
http://schemas.xmlsoap.org/ws/2004/09/transfer/Put	Supports WS-Transfer:Put
http://schemas.xmlsoap.org/ws/2004/09/transfer/Create	Supports WS-Transfer:Create
http://schemas.xmlsoap.org/ws/2004/09/transfer/Delete	Supports WS-Transfer>Delete
http://schemas.xmlsoap.org/ws/2005/06/management/Rename	Supports WS-Management Rename
http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate	Supports WS-Transfer:Enumerate
http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe	Supports WS-Eventing "Subscribe" operations, an <b>event source</b>

<a href="http://schemas.xmlsoap.org/ws/2005/06/management/fragmentAccess">http://schemas.xmlsoap.org/ws/2005/06/management/fragmentAccess</a>	Supports fragment-level access. Used in combination with other Action values.
<a href="http://schemas.xmlsoap.org/ws/2005/06/management/abstract">http://schemas.xmlsoap.org/ws/2005/06/management/abstract</a>	The catalog entry is only there to act as a parent for other linked entries. It is essentially a documentation entry and cannot be combined with other action types.
<a href="http://schemas.xmlsoap.org/ws/2005/06/management/methods">http://schemas.xmlsoap.org/ws/2005/06/management/methods</a>	Indicates the resource supports custom methods defined in WSDL
<a href="http://schemas.xmlsoap.org/ws/2005/06/management/event">http://schemas.xmlsoap.org/ws/2005/06/management/event</a>	The resource is an event
...vendor extensions...	Any specific operations or methods that vendors need to support

If WSDL is being used to publish vendor-specific methods, then the Operation entry is decorated with a WsdlPort attribute, a WsdlRef attribute and the Action elements refer to specific wsa:Action URIs representing specific operations ("methods") within that WSDL definition:

```
(1) <Operation WsdlPort="DiskOps"
(2)   WsdlRef="http://chemas.acme.com/2005/06/hardware/wsdl">
(3)
(4)   <Action> http://chemas.acme.com/2005/06/hardware/wsdl/reset </Action>
(5)   <Action> http://chemas.acme.com/2005/06/hardware/wsdl/format </Action>
(6)
(7) </Operation>
```

The Action may be repeated for each operation or method being published as a management operation.

## 2.12.2 SelectorSetRef

The SelectorSetRef element identifies which set of Selectors apply to this operation. More than one may apply:

```
(1) <Operation>
(2)   <Action>
(3)     http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
(4)   </Action>
(5)   <SelectorSetRef Name="s1"/>
(6) ...
```

Note that this is a *reference* to the separate element of the same name under the Access block which *defines* the selectors, their names, and types.

If the resource is accessed entirely through its ResourceURI and no selectors apply, then the SelectorSetRef element is omitted.

If more than one of several SelectorSetRef elements can apply, then they are each listed. These are not logically ANDed, but imply alternate SelectorSets; the client may choose any one of the specified SelectorSetRef elements to access the resource instance in a protocol operation, but not more than one.

The **Name** usage is arbitrary and only used within the scope of the Resource document for cross-referencing.

**CR 2.12.c:** *If Selectors are required to access the resource, the catalog entry MUST define the referenced SelectorSets and properly reference it from within all applicable Operation blocks.*

### 2.12.3 OptionSetRef

Similarly, the OptionSetRef element refers to a separately defined Options block (of the same name) under the Access block:

```
(1) <Operation>
(2)   <Action>
(3)     http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
(4)   </Action>
(5)   <SelectorSetRef Name="s1"/>
(6)   <OptionSetRef Name="opts2"/>
(7)   ...
```

As with Selectors, if more than one set of options applies, all the Options references are listed. These are not logically ANDed; the client may choose any one of the specified OptionSetRef elements during access to the resource.

The naming is arbitrary and only used within the scope of the Resource document for cross-referencing. If no Options apply or are available for this particular operation, the OptionSetRef element is omitted.

**CR 2.12.d:** *If Options are possible for a Resource during access, the catalog entry MUST define the OptionSet and properly reference it from within all applicable Operation blocks.*

### 2.12.4 SchemaRef

For operations other than deletions, the catalog may also specify the XML Schema namespace and XML QName of the item within the schema that applies to the operation, if applicable:

```

(1) <Operation>
(2)   <Action>
(3)     http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
(4)   </Action>
(5)   <SelectorSetRef Name="s1"/>
(6)   <SchemaRef xmlns:dsk="http://schemas.acme.com/2005/06/storage/disk">
(7)     dsk:DiskDefinition
(8)   </SchemaRef>
(9) ...

```

The above example states that a "Get" message, using the Selectors set s1 will return dsk:DiskDefinition elements, as defined in the namespace **http://schemas.acme.com/2005/06/storage/disk**.

The SchemaRef element may have an optional Location attribute which indicates the physical location of the schema body if it is not the same as its namespace URI:

```

(10)   <SchemaRef
(11)     Location="http://schemas.acme.com/2005/06/schemaList/st/dsk"
(12)     xmlns:dsk="http://schemas.acme.com/2005/06/storage/disk">
(13)     dsk:DiskDefinition
(14)   </SchemaRef>

```

The SchemaRef element may always be omitted.

## 2.12.5 *FilterDialect*

For fragment-level access, enumeration, and subscriptions, various "filter" languages may apply. The applicable filter dialects may be referenced, with the implication that they apply to the action defined in the Action block:

```

(1) <Operation>
(2)   <Action>
(3)     http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate
(4)   </Action>
(5)   <FilterDialect>
(6)     http://www.w3.org/TR/1999/REC-xpath-19991116
(7)   </FilterDialect>
(8) ...

```

FilterDialect can be repeated to indicate all the filter dialects that are applicable.

The above example indicates support for enumeration using XPath as a filter dialect. This element may be repeated once for each dialect or omitted.

This is primarily used with one of the two following Action URIs:

- http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate
- http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe

If the enumeration *requires* a filter in all cases (simple enumeration is not supported, then a `<FiltersRequired/>` element is added:

```
(9) <Operation>
(10)     <Action>
(11)         http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate
(12)     </Action>
(13)     <FilterDialect>
(14)         http://www.w3.org/TR/1999/REC-xpath-19991116
(15)     </FilterDialect>
(16)     <FiltersRequired/>
(17)     ...
```

...

## 2.12.6 *DeliveryModes*

If the resource Action URI is `http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe`, then the entry is an event source. In this case, one or more `DeliveryMode` elements must be present to indicate to the subscriber which event delivery modes are available:

```
(1) <Operation>
(2)     <Action>
(3)         http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe
(4)     </Action>
(5)     <FilterDialect>
(6)         http://www.w3.org/TR/1999/REC-xpath-19991116
(7)     </FilterDialect>
(8)     <DeliveryMode>
(9)         http://schemas.xmlsoap.org/ws/2005/06/management/Pull
(10)    </DeliveryMode>
(11)    <DeliveryMode>
(12)         http://schemas.xmlsoap.org/ws/2004/08/eventing/DeliveryModes/Push
(13)    </DeliveryMode>
(14)    ...
```

The above example documents an event source which supports XPath filtering and which can deliver events using one of two delivery modes.

**CR 2.12.e:** *If the resource Action URI is `http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe` then at least one `DeliveryMode` element MUST be present.*

---

## 2.13 SelectorSet

Multi-instanced resources require one or more Selector elements at the protocol level in order to fully identify a target. The Selectors are passed along with the ResourceURI to "select" the correct instance. There is no requirement that the values defined be part of the resource, only that they help to identify it. Selectors are not defined as a strict key mechanism in the sense that there must be no overlapping values or naming. Rather, selectors act more loosely, like queries, as long as a particular combination of Selectors results in identification of a single instance. That said, the concept of a "Key" is generally a good way to think about Selectors.

The definition and types of the Selector elements are located in one or more **SelectorSet** elements, each of which has a **Name** for cross-referencing purposes:

```
(1) <Access>
(2)   <Compliance> ... </Compliance>
(3)   <Operation> ... </Operation>
(4)
(5)   <SelectorSet Name="ByDrive">
(6)     <Selector Name="Drive" Type="xs:string">
(7)       The drive letter with colon, such as 'c:'
(8)     </Selector>
(9)   </SelectorSet>
(10)
(11)  <SelectorSet Name="ByVolume">
(12)    <Selector Name="Label" Type="xs:string">
(13)      The volume label
(14)    </Selector>
(15)  </SelectorSet>
(16)
```

The above declares two SelectorSets, one designed to access the disk by its drive letter ("ByDrive") and one designed to access it by its volume label ("ByVolume"). Note that each Selector has a Name (used in the WS-Management protocol) and a Type (which indicates the data type to use in the protocol message). The element content is a text description of what values are intended for use in the Selector. More than one Selector element may be defined in a SelectorSet block.

At runtime, all the Selector elements listed for a Selectors block **MUST** appear in a protocol message along with the ResourceURI in order to precisely identify the Resource instance. The **Name** is arbitrary and only used for cross-referencing the SelectorSet with the correct Actions within the Resource document.

In a typical case, one of these Selectors definitions is referenced by an Operation definition:

```
(17) <Access>
(18)   <Operation>
(19)     <Action>
```

```

(20)         http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
(21)         </Action>
(22)         <SelectorSetRef Name="ByDrive"/>
(23)     </Operation>
(24)     ...
(25)

```

The above example indicates that the "Get" operation for the disk requires a selector called "Drive" in the SOAP message because it requires the "ByDrive" SelectorSet. This SelectorSet contains a single Selector named **Drive** whose value is the drive letter with colon.

At runtime at the protocol level, all Selector elements for a SelectorSet block must appear in the protocol message along with the ResourceURI affixed to the address in the wsa:To block:

```

(26) <wsa:To> Network Address + ResourceURI </wsa:To>
(27) <wsman:SelectorSet>
(28)   <wsman:Selector Name="ByDrive"> C: </wsman:Selector>
(29) </wsman:SelectorSet>

```

**CR 2.13.a:** *For each Selector that can appear at the protocol level, there MUST be a definition establishing the name of the Selector and its type. If no selectors are required to access the entry (the ResourceURI represents a singleton entry), then the SelectorSet element is omitted.*

**CR 2.13.b:** *A Selector definition MAY provide descriptive text in the element describing how the Selector should be used. This text MUST be in the language identified by the **xml:lang** attribute in the Resource element.*

**CR 2.13.c:** *A Selector Type MUST consist of a simple base XML schema type imported from the standard XML schema namespace <http://www.w3.org/2001/XMLSchema> or a WS-Management EPR.*

In practice, all the Selector elements grouped under a single SelectorSet element must be present, and are logically ANDed. Since an operation may support more than one SelectorSet, the SelectorSet elements are logically ORed.

In essence, one SelectorSet is chosen, and all of its individual Selector elements are expected in the SOAP message. However, the client of an operation can choose between the sets, on a case-by-case basis, even for the same instance, as long as multiple SelectorSet elements are defined for that operation.

There is no requirement preventing Selector names or values from overlapping. As long as the combinatorial effect of a set of Selectors at the protocol level allows identification of the correct instance or set of instances for the protocol operation.

Note that while the all Selector values for a SelectorSet must be present in a protocol operation, the WS-Management specification states that it may short-circuit evaluating

all of the specified Selector values once it has unambiguously identified an instance (short-circuiting logic is expected).

---

## 2.14 Options

In many cases, operations require additional instructions or switches which modify the behavior of an operation. It is not appropriate to include these as Selector elements. Instead, they become Option elements at the protocol level in addition to Selector elements and the ResourceURI. As with Selectors, each option that is possible must be defined by name and value.

```
(1)    <Access>
(2)      <Compliance> ...
(3)      <Selectors>...
(4)      <OptionSet Name="MainOptions">
(5)        <Option Name="IncludeRedirected" Type="xs:boolean">
(6)          Include any redirected drives in the access scope.
(7)        </Option>
(8)      </OptionSet>
(9)      ...
```

Typically, Options are only used to modify the result of WS-Transfer or WS-Enumeration methods. If there is a custom method described in WSDL, the options typically become simple parameters to those methods.

**CR 2.14.a:** For each option that appears at the protocol level, there **MUST** be a definition establishing the name of the option and its type.

**CR 2.14.b:** An Option definition **MAY** provide descriptive text in the element describing how the option should be used and how it may relate to other options at the same scope. This text **MUST** be in the language identified by the **lang** attribute in the Resource element.

**CR 2.14.c:** An Option **MUST** consist of a simple base XML schema type imported from the standard XML schema namespace <http://www.w3.org/2001/XMLSchema>.

**CR 2.14.d:** Options **MUST NOT** be used to switch between alternate schemas for input or output. A new ResourceURI is the correct technique to do this.

**CR 2.14.e:** Options **MUST NOT** be used to simulate arbitrary methods over WS-Transfer or WS-Enumerate messages. The operation must have the same meaning without the option.

**CR 2.14.f:** Options **MUST NOT** be used to simulate simple parameters to custom methods defined in WSDL. This would mean that the WSDL definition was not self-sufficient to describe the method.

A typical use for Options is to "include expensive or optional properties in the result". Many XML schemas have optional elements, and often these are expensive to retrieve. Since the same schema can legally express both the short and long versions of the object, a hint is required to signal to the agent to proceed with the more expensive retrieval.

At runtime, only one OptionSet is active and used in the message for any given protocol operation.

---

## 2.15 Relationships

This section is used to describe relationships between catalog entries as well as the relationship of a catalog entry to the outside world. A relationship is a tuple of a relationship type, a role, and a pointer to the referenced entry.

As an introductory example, assume that the catalog entry needs to point to the vendor website for further information about the component that the catalog entry relates to:

```
(1) <Resource>
(2)   ...other
(3)
(4) <Relationships>
(5)
(6)   <Relationship Type="xml:QName" Role="xml:QName" Ref="xs:anyURI" />
(7)
(8) </Relationships>
```

The **Type** of a relationship (line 6) is an XML QName which identifies the relationship type, and **Ref** (line 8) points to the entry being related. An optional element **Role** element (another XML QName) indicates in what capacity the relationship exists, or "why" it is there.

**CR 2.15.a:** Any relationships between catalog entries *SHOULD* be described using this technique and *SHOULD NOT* rely solely on URI structuring to describe them.

**CR 2.15.b:** Relationships based on the URI structure *MAY* be used in addition to Relationship types.

### 2.15.1 Unary Relationships

For simple unary relationships to items outside the catalog space (such as to web sites), the Type is set to one of the standards from the following table, and the value of the Ref

attribute is the link. The Role is set to `wsmam:External`, indicating that the link is to some item outside of the WS-Management catalog space:

```
(9) <Relationships xmlns:wsmamcat="http://schemas.xmlsoap.org/ws/2005/06/wsmamcat"
(10) <Relationship Type="wsmamcat:relVendorURL" Role="wsmamcat:External"
(11)   Ref="http://www.acme.com/default.htm" />
(12)
(13) <Relationship Type="wsmamcat:relVendorEmail" Role="wsmamcat:External"
(14)   Ref="mailto:user@host?subject=WSManUserContact" />
(15) </Relationships>
```

For simple unary relationships that occur within a catalog, the *Role* can be omitted or set to `wsmamcat:Catalog`. This means that the relationship points to another catalog entry and the value of the *Ref* attribute is the *ResourceURI* of the other entry:

```
(16) <Relationships xmlns:wsmamcat="http://schemas.xmlsoap.org/ws/2005/06/wsmamcat"
(17)
(18) <Relationship Type="wsmamcat:relExtends" Role="wsmamcat:Catalog"
(19)   Ref="http://schemas.acme.com/2005/06/samples/baseStorageDevice" />
(20)
(21) </Relationships>
```

The above example states that the current catalog item extends the capabilities of the other item being referenced. The standard unary types are:

Relationship Type	Role	Meaning of URI
<code>wsmamcat:relVendorURL</code>	<code>wsmamcat:External</code>	A pointer to the vendor's documentation about the current component or catalog entry.
<code>wsmamcat:relExtends</code>	<code>wsmamcat:Internal</code>	Logically extends the capabilities described in another catalog entry.
<code>wsmamcat:relReplaces</code>	<code>wsmamcat:Internal</code>	Logically replaces an existing catalog entry, which is considered to be deprecated
<code>wsmamcat:relVendorEMail</code>	<code>wsmamcat:External</code>	Points to a e-mail address where the vendor can be reached for information.
<code>wsmamcat:relReplacedBy</code>	<code>wsmamcat:Internal</code>	A forward link to another catalog entry that replaces the current entry. The user should move to the new entry and use it instead.
<code>wsmamcat:relPublishesEventsTo</code>	<code>wsmamcat:Internal</code>	Indicates that the current Resource publishes events to some other Resource which

		supports the wse:Subscribe operation.
--	--	---------------------------------------

Any new arbitrary relationship may be defined by simply defining a new QName based on a new XML namespace.

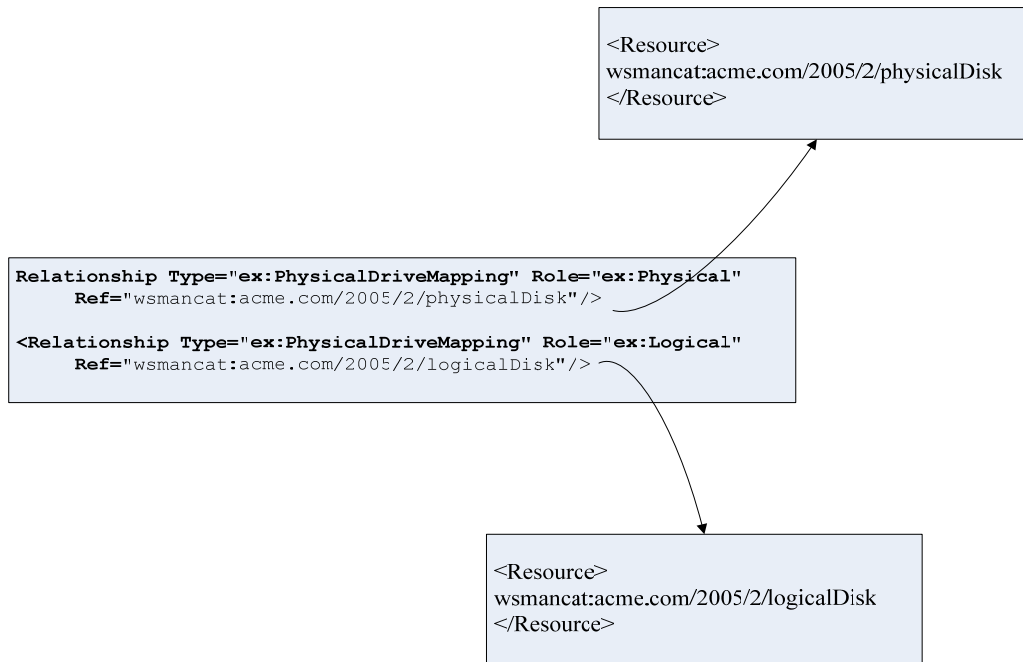
## 2.15.2 *Binary and N-Ary Relationships*

Binary relationships may be defined using two Relationship elements of the same Type to bind two items together. To distinguish between the two, different Role values are used.

For example, to describe a relationship between a physical and logical disk, references to each are required:

```
(1) <Relationships xmlns:ex="http://examples.org/2005/06/relationships"
(2)
(3)   <Relationship Type="ex:PhysicalDriveMapping" Role="ex:Physical"
(4)     Ref="wsmanecat:acme.com/2005/2/physicalDisk"/>
(5)
(6)   <Relationship Type="ex:PhysicalDriveMapping" Role="ex:Logical"
(7)     Ref="wsmanecat:acme.com/2005/2/logicalDisk"/>
(8)
```

This relationship means that the *current* catalog entry describes a mapping relationship between two other catalog entries. The example shows that physical drive objects are associated or mapped to logical drive objects:



All the relationships in the catalog are meant to provide metadata-level hints as to how catalog items relate to each other, but because the information is read-only and the catalog only contains type-level information, it does not attempt to describe specific relationships among instances or the "live objects". Rather, these relationships are to help tools discover how catalog items behave together.

Any number of user-defined relationships of any cardinality may be added by simply declaring XML namespaces and devising QNames for the types and Roles and by referencing the URI of the item to be related.

---

## 2.16 Open Content

The remainder of the catalog schema after the Relationships block is intentionally left for open content. Vendors may add any specific XML that is needed:

```

(1) <Resource
(2)   xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsman:acme.com/2005/2/physicalDisk"
(3)   xmlns:xs="http://www.w3.org/2001/XMLSchema"
(4)   >
(5)
(6) <ResourceURI>
(7)   wsman:mailservers.com/OurServer/perfdata/historical
(8) </ResourceURI>
(9)   ...
(10) <Access> ... </Access>
(11) <Relationships> ... </Relationships>

```

```
(12)
(13)   <AcmeInfo xmlns="http://schemas.acme.com/wsmancat/2005/06/extensions"
(14)       ...vendor-specific content
(15)   </AcmeInfo>
(16) </Resource>
```

Any extension must be namespace-qualified.

**CR 2.16.a:** *A vendor MAY add additional content under the Resource element as the last element, but this element MUST be XML namespace qualified.*

---

## 3 Events

The event model used by WS-Management separates the notion of an event source from the events themselves.

In the management world, events are sometimes provided by logs or logging devices, and usually many different components may contribute to the event log or stream. In other cases, only a single component or device publishes the events and there is no log. In some cases, a given stream or log of events is likely to contain various types of schemas.

If a resource is acting as an event source, it is marked with the wse:Subscribe Action URI:

```
(1) <Operation>
(2)   <Action>
(3)     http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe
(4)   </Action>
```

The declaration on line 3 asserts that the resource is acting as an event source and that wse:Subscribe may be used to access the content.

Note that another Action may be also declared, so that users can enumerate the log content independently of subscribing, if the event source is backed by a log:

```
(5) <Operation>
(6)   <Action>
(7)     http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe
(8)   </Action>
(9)   <Action>
(10)    http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate
(11)  </Action>
```

In this second case, the combination of the two simply implies that the event source is a "log" which can be enumerated. Similarly, if individual event log items can be identified by Selectors, then "Get" might be additionally supported.

As components are added to the catalog which can produce events, they may explicitly state their presence, schema, and which event source the events will be routed to for subscriptions.

It would be difficult to maintain a catalog if the event source Resource were updated or rewritten to accommodate new components and their events, so instead new components add new catalog entries with their events and *reference* the event source:

```
(12) <Operation>
(13)   <Action>
(14)     http://schemas.xmlsoap.org/ws/2005/06/management/event
(15)     <!-- This entry models an "event" -->
(16)   </Action>
(17)   <SchemaRef xmlns:ev="management.samples.org/2005/06/hardwareEvents"
(18)     ev:FanFailureEvent
(19)   </SchemaRef>
(20)   <SchemaRef xmlns:ev="management.samples.org/2005/06/hardwareEvents"
(21)     ev:OverheatedCPUEvent
(22)   </SchemaRef>
(23) </Operation>
(24)
(25) <Relationships>
(26)   <Relationship Type="wsman:relPublishesEventsTo
(27)     Ref="wsman:events.acme.com/2005/06/eventLog" />
(28)
```

Note that each distinct event possible is listed under the <Operation> element, which states that it is an "event" with any of the listed schemas. The actual source to use for subscriptions is indicated in the Relationships section of the entry, as shown on lines 25-27. The Reference element contains the catalog ResourceURI of the appropriate event source entry.

If a resource directly supports subscriptions, it can have both "Event" and "Subscribe" assertions as well as assertions for the events in the same catalog entry and a Relationship entry is not needed:

```
(29) <Operation>
(30)   <Action>
(31)     http://schemas.xmlsoap.org/ws/2005/06/management/event
(32)   </Action>
(33)   <Action> http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe </Action>
(34)
(35)   <SchemaRef xmlns:ev="management.samples.org/2005/06/hardwareEvents"
(36)     ev:FanFailureEvent
(37)   </SchemaRef>
```

```
(38)      <SchemaRef xmlns:ev="management.samples.org/2005/06/hardwareEvents"
(39)          ev:OverheatedCPUEvent
(40)      </SchemaRef>
(41) </Operation>
```

When a wse:Subscribe operation is performed, it is sent to the owning ResourceURI. If that ResourceURI is as shown above and can provide more than one event type, then the desired events are identified using the event "Filter" (the wsen:Filter element).

Typically, the Filter is an XPath query which can select FanFailureEvent or OverheatedCPUEvent or both.

However, there is no requirement that XPath be the mechanism used for filtering. New Dialects specific to a service implementation are permitted and desirable if XPath is too heavyweight for a resource-constrained implementation. Alternately, SelectorSets may be used to partition the event space for a given event source, and in many cases no Filtering would be required.

---

## 4 WSDL Usage

---

### 4.1 General

WSDL 1.1 is used to formally describe the resource from a SOAP and Web services perspective. The WSDL document for a resource defines its addressing information and which of the protocol operations are compatible with the resource.

WSDL is optional, but highly recommended, as the Resource format alone cannot fully describe how to construct valid messages for WS-Management operations, and WSDL is *required* in cases where custom methods are exported by the resource.

WSDL may be used exclusively for the custom methods exported by the resource, or it may contain definitions for all of the applicable WS-Management operations. This section describes how to define the WSDL so that all of the standard WS-Management operations (such as WS-Transfer:Get, Put, and WS-Enumeration:Enumerate, etc.) can be added to the WSDL.

In general, to facilitate the construction of management clients, it is recommended that the WSDL definition contain *all* of the relevant message definitions, including the ones from the WS-Management specification. In this way, the WSDL document becomes a single-source definition for the management operations for the specified resource.

---

### 4.2 Primary Issues

When constructing WSDL that is compatible with WS-Management, the following syntax components require special treatment. Apart from that, the WSDL is normal:

- (4) wsa:Action URI
- (5) wsman:ResourceURI
- (6) wsman:SelectorSet
- (7) wsman:OptionSet

---

## 4.3 wsa:Action URIs

If the WSDL is describing custom methods which do not overlap with the operations defined for WS-Management (such as wxf:Get, wxf:Set, wse:Subscribe, etc.), then the designer selects any applicable wsa:Action URI for the method.

If the designer is indicating that a standard WS-Management operation is being modeled, such as WS-Transfer:Get, then the wsa:Action URI for the operation *must* be the same as defined within WS-Management and its component specifications.

This means that when defining the wsdl:portType

```
(1) <wsdl:portType
(2)     name="SystemProcess"
(3)
(4) <!--Standard Get operation-->
(5) <wsdl:operation name="Get">
(6)     <wsdl:input message="tns:SystemProcess.Get"
(7)         wsa:Action="http://schemas.xmlsoap.org/ws/2004/08/transfer/Get" />
(8)     <wsdl:output message="tns:SystemProcess.GetResponse"
(9)         wsa:Action="http://schemas.xmlsoap.org/ws/2004/08/transfer/GetResponse" />
(10)     </wsdl:operation>
(11)     ...
```

All of the messages need to be modeled, including any response messages. The designer should consult the WSDL documents for the underlying specifications, such as the WS-Transfer, or WS-Enumeration WSDL definitions.

---

## 4.4 WS-Management Endpoint Reference Elements

To reference a WS-Management endpoint, a wsman:ResourceURI is required, and optionally a wsman:SelectorSet and wsman:OptionSet may appear. WSDL message parts are defined for these:

```
(1) <!--Single WSDL Message to contain all keys (reference params)-->
(2) <wsdl:message name="SelHeader">
(3)     <wsdl:part name="selectors" element="wsman:SelectorSet"/>
(4) </wsdl:message>
```

```

(5)
(6) <!--Single WSDL Message to contain ResourceURI (reference prop) -->
(7) <wsdl:message name="ResourceHeader">
(8)   <wsdl:part name="resourceURI" element="wsman:ResourceURI"/>
(9) </wsdl:message>
(10)
(11) <!--Single WSDL Message to contain Options (reference params) -->
(12) <wsdl:message name="OptionsHeader">
(13)   <wsdl:part name="options" element="wsman:OptionSet"/>
(14) </wsdl:message>

```

Then, when defining bindings, the `wsa:Action` URI and WS-Management SOAP headers can be combined into a wire-compatible definition:

```

(15)   ...
(16)   <wsdl:binding name="..." type="...">
(17)     <wsdlsoap11:binding style="document"
(18)       transport="http://schemas.xmlsoap.org/soap/http" />
(19)
(20)   <!--Standard Get operation-->
(21)   <wsdl:operation name="Get"
(22)     soapAction="http://schemas.xmlsoap.org/ws/2004/08/transfer/Get">
(23)     <wsdlsoap11:operation
(24)       soapAction="http://schemas.xmlsoap.org/ws/2004/08/transfer/Get"
(25)       style="document" />
(26)     <wsdl:input>
(27)       <wsdlsoap11:body use="literal" />
(28)       <wsdlsoap11:header use="literal"
(29)         message="..." part="resourceURI"/>
(30)       ...
(31)     </wsdl:input>
(32)     <wsdl:output>
(33)       <wsdlsoap11:body use="literal" />
(34)     </wsdl:output>
(35)   </wsdl:operation>

```

---

## 5 Retrieving Metadata

---

### 5.1 Introduction

This section describes how resource metadata is actually retrieved by client applications. This is done using WS-Management itself, specifically using WS-Transfer:Get and WS-Enumeration.

It is important to be able to systematically enumerate the available resources for a given agent or agent/system combination. For a given service, the catalog can be reached by accessing a standard ResourceURI:

### **wsman:system/catalog/2005/06/Catalog**

If a **WS-Enumeration:Enumerate** operation is performed against this URI, then an enumeration of the entire catalog will occur. Line 12 illustrates this URI being used in an enumerate request:

```
(1) <env:Envelope
(2)   xmlns:s="http://www.w3.org/2003/05/soap-envelope"
(3)   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
(4)   xmlns:wsman="http://schemas.xmlsoap.org/ws/2004/10/management"
(5)   xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
(6)   >
(7)
(8)   <env:Header>
(9)     <wsa:To>
(10)    http://1.2.3.4/wsman
(11)   </wsa:To>
(12)   <wsman:ResourceURI> wsman:system/catalog/2005/06/Catalog </wsman:ResourceURI>
(13)   <wsa:ReplyTo>
(14)     http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
(15)   </wsa:ReplyTo>
(16)   <wsa:Action env:mustUnderstand="true">
(17)     http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate
(18)   </wsa:Action>
(19)   <wsa:MessageID env:mustUnderstand="true">
(20)     uuid:d9726315-bc91-430b-9ed8-ce5ffb858a21
(21)   </wsa:MessageID>
(22)   ...
(23)
```

Each element that is retrieved will be of type <Resource> as described in this document and these will be the items retrieved in the WS-Enumeration:Pull operation.

The resulting responses in the WS-Enumeration:PullResponse place the <Resource> entries described in this specification back-to-back:

```
(24) <wsen:PullResponse>
(25)   <wsen:EnumerationContext> 0xAF0021 </wsen:EnumerationContext>
(26)   <wsen:Items>
(27)     <Resource> ... </Resource>
(28)     <Resource> ... </Resource>
```

**CR 5.1.a:** Implementations *MAY* support XPath or other filtering dialects over the catalog, but this is *NOT REQUIRED*.

**CR 5.1.b:** Implementations *SHOULD* support a simple enumeration of the catalog or else point to a location where such an enumeration can occur via a SOAP fault using the techniques in 6.2 6.3.

If a **WS-Transfer:Get** is performed against this URI, then a Selector must be specified indicating which catalog entry is required. This Selector takes on the value of the ResourceURI of the catalog entry which is desired. Note the use of ResourceURI for the catalog itself (in line 8) and the targeted catalog Resource in the SelectorSet (line 24):

```
(1) <s:Envelope
(2)   xmlns:s="http://www.w3.org/2003/05/soap-envelope"
(3)   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
(4)   xmlns:wsman="http://schemas.xmlsoap.org/ws/2004/10/management">
(5)
(6) <s:Header>
(7)   <wsa:To>
(8)   http://1.2.3.4/wsman
(9)   </wsa:To>
(10)  <wsman:ResourceURI> wsman:system/catalog/2005/06/Catalog </wsman:ResourceURI>
(11)  <wsa:ReplyTo>
(12)    http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
(13)  </wsa:ReplyTo>
(14)
(15)  <wsa:Action env:mustUnderstand="true">
(16)    http://schemas.xmlsoap.org/ws/2004/01/transfer/Get
(17)  </wsa:Action>
(18)
(19)  <wsa:MessageID env:mustUnderstand="true">
(20)    uuid:d9726315-bc91-430b-9ed8-ce5ffb858a87
(21)  </wsa:MessageID>
(22)
(23)  <wsman:SelectorSet>
(24)    <wsman:Selector Name="ResourceURI">
(25)      wsman:acme.com/2005/06/physDisk
(26)    </wsman:Selector>
(27)  </wsman:SelectorSet>
(28)
(29)  </s:Header>
(30)  <s:Body/>
(31)
(32) </s:Envelope>
```

**CR 5.1.c:** Implementations *SHOULD* support the ability to retrieve individual entries using WS-Transfer:Get. If so, the ResourceURI *MUST* refer to a specific catalog

format, and the SelectorSet MUST refer to the resource for which the catalog entry is desired..

The resulting wxf:GetResponse contains a single catalog entry:

```
(33) <s:Envelope
(34)     xmlns:s="http://www.w3.org/2003/05/soap-envelope"
(35)     xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
(36)     xmlns:wsmn="http://schemas.xmlsoap.org/ws/2004/10/management">
(37)   <s:Header>
(38)     <wsa:To>
(39)       http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
(40)     </wsa:To>
(41)
(42)     <wsa:Action env:mustUnderstand="true">
(43)       http://schemas.xmlsoap.org/ws/2004/01/transfer/GetResponse
(44)     </wsa:Action>
(45)
(46)     <wsa:MessageID env:mustUnderstand="true">
(47)       uuid:d9726315-bc91-430b-9ed8-ce5ffb858a88
(48)     </wsa:MessageID>
(49)
(50)     <wsa:RelatesTo>
(51)       uuid:d9726315-bc91-430b-9ed8-ce5ffb858a87
(52)     </wsa:RelatesTo>
(53)   </s:Header>
(54)
(55)   <s:Body>
(56)     <Resource
(57)       xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat"
(58)       xmlns:xs="http://www.w3.org/2001/XMLSchema"
(59)       xml:lang="en-us" >
(60)       <ResourceURI>
(61)         wsmn:acme.com/2005/06/physDisk
(62)       </ResourceURI>
(63)       ...rest of catalog entry...
(64)     </Resource>
(65)   </s:Body>
(66) </s:Envelope>
```

---

## 5.2 Subsequent Catalog Versions and Schemas

Because the catalog may evolve over time, the ResourceURI above may eventually coexist with other URIs for later versions of the catalog specification or for other metadata types that are different than the current one.

The following well-known ResourceURI refers to the usage defined in this specification:

**wsman:system/catalog/2005/06/Catalog**

However, in subsequent versions, the Resource format may be republished with a new XML schema at some point in the future. Then clients may be expected to know about a new URI which will return this format, for example:

**wsman:system/catalog/2007/09/Catalog**

In this way, upgrades to the metadata formats can occur while previous ones are still being supported.

It may be necessary to query an agent to determine which versions and which URIs are available. In this case, see section 6.3.

**CR 5.2.a:** Implementations which use **wsman:system/catalog/2005/06/Catalog** as a ResourceURI MUST return resource elements from the schema namespace <http://schemas.xmlsoap.org/ws/2005/06/wsmancat>.

---

## 5.3 Catalog Location and Indirection

The catalog may not reside at the same physical address as the agent. Similarly, the client may not initially know which versions of the catalog are available under which URIs for a given agent.

When attempting to access the standard catalog URI, the service may return a **wsmancat:MetadataRedirect** fault, which specifies the actual address of the catalog. The fault may contain either a SOAP-aware address using WS-Management itself, or it may contain an internet address, such as an HTTP, HTTPS, or FTP address.

Systems which share catalogs with other systems may use this technique to point to the shared catalog.

The fault is encoded in the same way as other WS-Management faults and has the following content:

<b>Fault Subcode</b>	wsmancat:MetadataRedirect
<b>Code</b>	s:Receiver
<b>Reason</b>	The catalog was not available at the specified location, but may be available at other locations.
<b>Detail</b>	<env:Detail>

	<pre> &lt;wsa:EndpointReference&gt; ...SOAP address ... &lt;/wsa:EndpointReference&gt; &lt;wsa:EndpointReference&gt; ...SOAP address ... &lt;/wsa:EndpointReference&gt; &lt;wsman:URL&gt; http address &lt;/wsman:URL&gt; &lt;wsman:URL&gt; https address &lt;/wsman:URL&gt; &lt;wsman:URL&gt; ftp address &lt;/wsman:URL&gt; ...etc. &lt;/env:Detail&gt; </pre>
<b>Comments</b>	This is returned whenever the catalog is not resident at the standard ResourceURI.
<b>Applicability</b>	wxf:Get, wsen:Enumerate
<b>Remedy</b>	Client may extract one of the addresses in the Detail element.

**CR 5.3.a:** Any agent implementation which does not have the catalog resident **MUST** support the above redirection fault to point to its catalog.

**CR 5.3.b:** If no catalog exists, the Detail element should contain an env:Text field explaining the absence of the catalog.

If the catalog is accessed by some other means than WS-Management, such as a plain HTTP POST request, then the Resource elements are wrapped in a single XML <Catalog> wrapper:

```

(1) <Catalog xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat">
(2)   <Resource> ... </Resource>
(3)   <Resource> ... </Resource>
(4)   <Resource> ... </Resource>
(5)   <Resource> ... </Resource>
(6)   ...etc.
(7) </Catalog>

```

---

## 5.4 Localized Metadata

Localization of metadata is done within the catalog agent implementation. When metadata is requested, the agent will attempt to ensure that the metadata returned matches the locale requested by the client. The entire <Resource> definition for a catalog entry is defined in one language using RFC 3066 language codes with the **xml:lang** attribute as seen in line 4:

```

(1) <Resource
(2)   xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat "
(3)   xmlns:xs="http://www.w3.org/2001/XMLSchema"
(4)   lang="en-us"
(5)   >
(6)
(7) </Resource>

```

It is up to the implementation to compose or retrieve catalog entries in the correct locale and replace any descriptive elements (such as <Notes> blocks) with localized equivalents and to indicate the locale by changing the **lang** attribute to match:

```

(8) <Resource
(9)   xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat "
(10)  xmlns:xs="http://www.w3.org/2001/XMLSchema"
(11)  lang="fr"
(12)  >
(13)  <ResourceURI> wsman:microsoft.com/os/version </ResourceURI>
(14)  <Version> 1.0 </Version>
(15)  <Notes>
(16)    Pour obtenir la version et
(17)    des informations du système installé de Windows
(18)  </Notes>
(19)  ...

```

The implementation may keep localized catalog entries or may localize them in real-time using string substitution techniques.

The technique for retrieving and composing the localized document is agent-specific. Most implementations will have a substitution string mechanism in descriptive fields and use a text processor to fill in the language-specific portions from external localization tables:

```

(20) <Resource
(21)   xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat "
(22)   xmlns:xs="http://www.w3.org/2001/XMLSchema"
(23)   xml:lang="$(lang)"
(24)   >
(25)   <ResourceURI> wsman:microsoft.com/os/version </ResourceURI>
(26)   <Version> 1.0 </Version>
(27)   <Notes>
(28)     $(localized_notes_1)
(29)   </Notes>
(30)

```

Note that from the perspective of a WS-Management service, the catalog localization happens in real-time. In some environments, this may present a problem (such as the Web). In order to identify different versions of the same catalog in different languages,

the # symbol can be added to the ResourceURI followed by an RFC 3066 language code:

```
(31) http://schemas.acme.com/samples/2005/06/systemCatalog#en-us
```

---

## 5.5 Retrieving XML Schema and WSDL

The catalog implementation may optionally support the retrieval of XML schema definitions and WSDL documents. This is done in the same way as for retrieving catalog entries as covered in 5.1.

To retrieve XML schema documents, the following ResourceURI is used

**wsman:system/catalog/2005/06/XMLSchema**

and the wsman:SelectorSet must have a wsman:Selector with Name="XMLSchema" whose value is the XML namespace of the desired schema. Only wxf:Get is supported as an operation. Enumeration of available schemas is not supported.

To retrieve a WSDL definition, the following ResourceURI is used

**wsman:system/catalog/2005/06/WSDL1.1**

and the wsman:SelectorSet must have a wsman:Selector with Name="WSDL1.1" whose value is the WSDL URI. Only wxf:Get is supported as an operation. The returned WSDL must conform to the WSDL 1.1 definition.

If the requested Schema or WSDL document cannot be found, a wsman:MetadataRedirect fault may be returned as defined in 5.3 if the service does know the external location of the requested document.

---

## 6 Appendix B : Resource XSD

```
<xs:schema
  targetNamespace="http://schemas.xmlsoap.org/ws/2005/06/wsmancat "
  xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat "
  xmlns:tns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat "
  xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema "
```

```

    elementFormDefault="qualified"
  >

<xs:complexType name="RelationshipType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="Type" type="xs:QName" use="required"/>
      <xs:attribute name="Role" type="xs:QName" use="optional"/>
      <xs:attribute name="Ref" type="xs:anyURI" use="required"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="SelectorType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="Name" type="xs:token" use="required"/>
      <xs:attribute name="Type" type="xs:QName" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="OptionType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="Name" type="xs:token" use="required"/>
      <xs:attribute name="Type" type="xs:QName" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="SelectorSetType">
  <xs:sequence>
    <xs:element name="Selector" type="tns:SelectorType"
      minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="Name" type="xs:token"/>
</xs:complexType>

<xs:complexType name="OptionSetType">
  <xs:sequence>

```

```

    <xs:element name="Option" type="tns:OptionType"
      minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="Name" type="xs:token" />
</xs:complexType>

<xs:complexType name="RelationshipListType">
  <xs:sequence>
    <xs:element name="Relationship" type="tns:RelationshipType"
      minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="KeywordListType">
  <xs:sequence>
    <xs:element name="Keyword" type="xs:QName" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="SimpleRefType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="Name" type="xs:token" use="required" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="SchemaRefType">
  <xs:simpleContent>
    <xs:extension base="xs:QName">
      <xs:attribute name="Location" type="xs:anyURI" use="optional" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="OperationType">
  <xs:sequence>
    <xs:element name="Action" type="xs:anyURI" minOccurs="1" maxOccurs="unbounded" />
    <xs:element name="SelectorSetRef" type="tns:SimpleRefType"
      minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="OptionSetRef" type="tns:SimpleRefType"

```

```

        minOccurs="0" maxOccurs="unbounded" />
<xs:element name="SchemaRef" type="tns:SchemaRefType"
    minOccurs="0" maxOccurs="unbounded" />
<xs:element name="FilterDialect" type="xs:anyURI"
    minOccurs="0" maxOccurs="unbounded" />
<xs:element name="DeliveryMode" type="xs:anyURI"
    minOccurs="0" maxOccurs="unbounded" />
</xs:sequence>
<xs:attribute name="WsdPort" type="xs:token" use="optional" />
<xs:attribute name="WsdRef" type="xs:anyURI" use="optional" />
<xs:attribute name="WsdLocation" type="xs:anyURI" use="optional" />
</xs:complexType>

<xs:complexType name="AccessType">
    <xs:sequence>
        <xs:element name="Compliance" type="xs:anyURI"
            minOccurs="1" maxOccurs="unbounded" />
        <xs:element name="Operation" type="tns:OperationType"
            minOccurs="1" maxOccurs="unbounded" />
        <xs:element name="SelectorSet" type="tns:SelectorSetType"
            minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="OptionSet" type="tns:OptionSetType"
            minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>

<xs:complexType name="ResourceType">
    <xs:sequence>
        <xs:element name="ResourceURI" type="xs:anyURI" minOccurs="1" maxOccurs="1" />
        <xs:element name="Notes" type="xs:string" minOccurs="0" maxOccurs="1" />
        <xs:element name="Vendor" type="xs:string" minOccurs="0" />
        <xs:element name="DisplayName" type="xs:string" minOccurs="0" />
        <xs:element name="Keywords" type="tns:KeywordListType" minOccurs="0" />
        <xs:element name="Access" type="tns:AccessType" minOccurs="1"
            maxOccurs="unbounded" />
        <xs:element name="Relationships" type="tns:RelationshipListType"
            minOccurs="0" maxOccurs="1" />
        <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="lang" type="xs:language" use="optional" />
</xs:complexType>

```

```

<xs:element name="Resource" type="tns:ResourceType" />
<xs:element name="Catalog">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Resource" type="tns:ResourceType"
        minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>

```

---

## 7 Appendix C : Example Catalog Entries

---

### 7.1 Minimal

This is a simple, minimal catalog entry for a simple read-only device with two operations, wxf:Get and wsen:Enumerate:

```

(1) <Resource
(2)   xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat"
(3)   xmlns:wsmancat="http://schemas.xmlsoap.org/ws/2005/06/wsmancat"
(4)   xmlns:xs="http://www.w3.org/2001/XMLSchema"
(5)   lang="en-us">
(6)
(7)   <ResourceURI> http://schemas.acme.com/2005/06/samples/physicalDisk </ResourceURI>
(8)   <Vendor> Acme Corporation </Vendor>
(9)
(10)  <Access>
(11)    <Compliance> http://schemas.xmlsoap.org/ws/2005/06/management </Compliance>
(12)
(13)    <Operation>
(14)      <Action>
(15)        http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
(16)      </Action>
(17)      <SelectorSetRef Name="s1" />
(18)    </Operation>
(19)
(20)    <Operation>
(21)      <Action>
(22)        http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate

```

```

(23)         </Action>
(24)         </Operation>
(25)
(26)         <SelectorSet Name="s1">
(27)             <Selector Name="LUN" Type="xs:unsignedInt">
(28)                 The logical unit number of the disk </Selector>
(29)             </SelectorSet>
(30)         </Access>
(31)     </Resource>

```

---

## 7.2 More Complex Example

```

(1) <Resource
(2)     xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat "
(3)     xmlns:wsmancat="http://schemas.xmlsoap.org/ws/2005/06/wsmancat "
(4)
(5)     xmlns:xs="http://www.w3.org/2001/XMLSchema "
(6)     lang="en-us">
(7)
(8)     <ResourceURI> http://schemas.acme.com/2005/06/samples/physicalDisk </ResourceURI>
(9)     <Notes> Describes access to a physical disk unit </Notes>
(10)    <Vendor> Acme Corporation </Vendor>
(11)    <DisplayName> ACME Super Disk </DisplayName>
(12)
(13)    <Keywords xmlns:acme="http://keywords.acme.com/2005/06/hardware">
(14)        <Keyword>acme:Disk</Keyword>
(15)        <Keyword>acme:Storage</Keyword>
(16)    </Keywords>
(17)
(18)    <Access>
(19)        <Compliance>
(20)            http://schemas.xmlsoap.org/ws/2005/06/management
(21)        </Compliance>
(22)
(23)    <Operation>
(24)        <Action>
(25)            http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
(26)        </Action>
(27)        <SelectorSetRef Name="s1"/>
(28)        <SchemaRef xmlns:ac="schemas.acme.com/2005/06/hardware">
(29)            ac:PhysDisk
(30)        </SchemaRef>
(31)    </Operation>
(32)

```

```

(33)     <Operation>
(34)         <Action>
(35)             http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate
(36)         </Action>
(37)         <OptionSetRef Name="osl" />
(38)         <SchemaRef Location="foo"
(39)             xmlns:ac="schemas.acme.com/2005/06/hardware">
(40)             ac:PhysDisk
(41)         </SchemaRef>
(42)     </Operation>
(43)
(44)     <Operation WsdlPort="DiskOps"
(45)         WsdlRef="http://chemas.acme.com/2005/06/hardware/wsdl">
(46)         <Action> http://chemas.acme.com/2005/06/hardware/wsdl/reset </Action>
(47)         <Action> http://chemas.acme.com/2005/06/hardware/wsdl/format </Action>
(48)     </Operation>
(49)
(50)     <SelectorSet Name="sl">
(51)         <Selector Name="LUN" Type="xs:unsignedInt">
(52)             The logical unit number of the disk </Selector>
(53)     </SelectorSet>
(54)
(55)     <OptionSet Name="osl">
(56)         <Option Name="Verbosity" Type="xs:unsignedInt">
(57)             Controls verbosity of output 0 (least) to 10 (most) </Option>
(58)     </OptionSet>
(59) </Access>
(60)
(61)     <Relationships>
(62)         <Relationship Type="wsmancat:VendorURL" Role="wsmancat:External"
(63)             Ref="http://www.acme.com" />
(64)     </Relationships>
(65)
(66) </Resource>

```

---

## 7.3 Event and Event Source Example

The following is an example of a resource which raises events:

```

(1) <Resource
(2)     xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsmancat"
(3)     xmlns:wsmancat="http://schemas.xmlsoap.org/ws/2005/06/wsmancat"
(4)     xmlns:xs="http://www.w3.org/2001/XMLSchema"
(5)     lang="en-us">
(6)

```

```

(7)    <ResourceURI> http://schemas.acme.com/2005/06/samples/physicalDisk </ResourceURI>
(8)    <Notes> Describes events from a physical disk unit </Notes>
(9)    <Vendor> Acme Corporation </Vendor>
(10)   <DisplayName> ACME Super Disk Events </DisplayName>
(11)
(12)   <Keywords xmlns:acme="http://keywords.acme.com/2005/06/hardware">
(13)     <Keyword>acme:DiskEvents</Keyword>
(14)     <Keyword>acme:StorageEvents</Keyword>
(15)   </Keywords>
(16)
(17)   <Access>
(18)     <Compliance> http://schemas.xmlsoap.org/ws/2005/06/management </Compliance>
(19)     <Operation>
(20)       <Action>
(21)         http://schemas.xmlsoap.org/ws/2005/06/management/event
(22)       </Action>
(23)       <SchemaRef xmlns:ac="schemas.acme.com/2005/06/hardware">
(24)         ac:LowDiskSpace
(25)       </SchemaRef>
(26)       <SchemaRef xmlns:ac="schemas.acme.com/2005/06/hardware">
(27)         ac:DiskTemperature
(28)       </SchemaRef>
(29)       <SchemaRef xmlns:ac="schemas.acme.com/2005/06/hardware">
(30)         ac:ExcessiveHeadRetries
(31)       </SchemaRef>
(32)     </Operation>
(33)   </Access>
(34)
(35)   <Relationships>
(36)     <Relationship Type="wsman:relPublishesEventsTo"
(37)       Ref="wsman:events.acme.com/2005/06/eventLog"/>
(38)   </Relationships>
(39) </Resource>

```

And here is the associated event source that the above publishes into. Note that the event source allows both subscribe and enumeration using the same filter dialect:

```

(40) <Resource
(41)   xmlns="http://schemas.xmlsoap.org/ws/2005/06/wsman"
(42)   xmlns:wsman="http://schemas.xmlsoap.org/ws/2005/06/wsman"
(43)   xmlns:xs="http://www.w3.org/2001/XMLSchema"
(44)   lang="en-us">
(45)
(46)   <ResourceURI>
(47)     http://schemas.acme.com/2005/06/samples/physicalDisk

```

```
(48)      </ResourceURI>
(49)      <Notes> Describes events from a physical disk unit </Notes>
(50)      <Vendor> Acme Corporation </Vendor>
(51)      <DisplayName> ACME Super Disk Events </DisplayName>
(52)
(53)      <Keywords xmlns:acme="http://keywords.acme.com/2005/06/hardware">
(54)          <Keyword>acme:DiskEvents</Keyword>
(55)          <Keyword>acme:StorageEvents</Keyword>
(56)      </Keywords>
(57)
(58)      <Access>
(59)          <Compliance> http://schemas.xmlsoap.org/ws/2005/06/management </Compliance>
(60)          <Operation>
(61)              <Action>
(62)                  http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe
(63)              </Action>
(64)              <Action>
(65)                  http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate
(66)              </Action>
(67)              <FilterDialect>
(68)                  http://www.w3.org/TR/1999/REC-xpath-19991116
(69)              </FilterDialect>
(70)              <DeliveryMode>
(71)                  http://schemas.xmlsoap.org/ws/2004/08/eventing/DeliveryModes/Push
(72)              </DeliveryMode>
(73)          </Operation>
(74)      </Access>
(75) </Resource>
```

---

## 8 References

### [WS-Management]

R. McCollum, et al.,

<http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-management.pdf>

### [SOAP 1.2]

M. Gudgin, et al, "[SOAP Version 1.2 Part 1: Messaging Framework](#)," June 2003.

### [WS-Addressing]

D. Box et al, "[Web Services Addressing \(WS-Addressing\)](#)," August 2004

### [WS-Enumeration]

J. Alexander et al, "[Web Services Enumeration \(WS-Enumeration\)](#)," September 2004

### [WS-Eventing]

D. Box et al, "[Web Services Eventing \(WS-Eventing\)](#)," August 2004

**[WS-Transfer]**

J. Alexander et al, "[Web Services Transfer \(WS-Transfer\)](#)," September 2004

**[WSDL 1.1]**

E. Christensen et al, "[Web Services Description Language \(WSDL\) 1.1](#)," March 2001.

**[XML Schema, Part 1]**

H. Thompson et al, "[XML Schema Part 1: Structures](#)," May 2001.

**[XML Schema, Part 2]**

P. Biron et al, "[XML Schema Part 2: Datatypes](#)," May 2001.

**[RFC 2396] Uniform Resource Identifiers (URI) : Generic Syntax**

<http://www.ietf.org/rfc/rfc2396.txt>

**[RFC 3066] Tags for the Identification of Languages**

<http://www.ietf.org/rfc/rfc3066.txt>