Web Services Resource Catalog (WS-RC)

Version 1.0

Authors

Alexander Nosov, Microsoft Corporation
Andrew Hately, IBM
Brian Reistad (Editor), Microsoft Corporation
Bryan Murray, HP
Doug Davis, IBM
Heather Kreger, IBM
Peter Niblett, IBM
Raymond McCollum (Editor), Microsoft Corporation
Vijay Tewari, Intel Corporation
Vishwa Kumbalimutt, Microsoft Corporation
William Vambenepe, HP

Copyright Notice

(c) 2006-2007 Hewlett-Packard Development Company (HP), Intel Corporation, International Business Machines Corporation (IBM), and Microsoft Corporation. All rights reserved.

Permission to copy and display the "Web Services Resource Catalog" Specification, in any medium without fee or royalty is hereby granted, provided that you include the following on ALL copies of the "Web Services Resource Catalog" Specification, or portions thereof, that you make:

1. A link or URL to the "Web Services Resource Catalog" Specification at this location: http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog.

2. The copyright notice as shown in the "Web Services Resource Catalog" Specification.

Hewlett-Packard Development Company (HP), Intel Corporation, International Business Machines Corporation (IBM), and Microsoft Corporation (collectively, the "Authors") each agree to grant you a royalty-free license, under reasonable, non-discriminatory terms and conditions to their respective patents that they deem necessary to implement the "Web Services Resource Catalog" Specification.

THE "WEB SERVICES RESOURCE CATALOG" SPECIFICATION IS PROVIDED "AS IS," AND THE AUTHORS 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 "WEB SERVICES RESOURCE CATALOG" 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 AUTHORS 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 "WEB SERVICES RESOURCE CATALOG" SPECIFICATION.

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

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

Abstract

This specification defines a catalog for organizing and classifying management resources. The catalog is intended to advertise resources accessible via WS-Management, WSDM and other management specifications including their underlying protocols. This specification can be composed with other Web service data description specifications.

Status

This specification is an initial draft. It is likely to change and there is no guarantee of compatibility between this version and subsequent versions. As a result, it should only be used for information, feedback and experimentation.

Table of Contents

1. Introduction

Applications that need to locate and retrieve addressing information about resources need a service that offers this information in a systematic manner.

This specification describes an XML document format that effectively supports the description of resources and their associated metadata in a model-neutral manner. Bindings for specific data-models can provide additional rules on how to advertise those resources in the catalog as well as include additional structure in extensibility points.

1.1 Requirements

This specification meets the following requirements.

a) It should define an approach to cataloging resources available to Web services clients that is data-model neutral. It should allow a single catalog to contain resources with different data models but allow clients to perform some data model-independent processing.

b) It should enable clients to discover resources based on certain search criteria, e.g. "all resources related to network management", "all resources that support event subscriptions" etc.

c) It should allow inclusion of information relevant to the type of the resource represented such as XML schema, WSDL, access protocols, eventing capabilities, etc. It should also support the ability to include subsets of resource data for discovery.

d) It should define a mechanism to link resources using relationships. It may define a few such relationship types but should allow for data model-specific relationship types.

e) It should support catalogs that are produced dynamically from other services, such as servers implementing the Common Information Model, as well as catalogs published as an XML file on a Web site or file system.

f) It should provide references to endpoints representing resources using the appropriate addressing technique such as URLs, WS-Addressing Recommendation, WS-Addressing W3C Member Submission, etc.

g) It should define extensibility points for currently unanticipated scenarios.

1.2 Non-Requirements

This specification does not intend to meet the following requirements:

a) Network discovery of an initial catalog address

b) Access protocols for retrieving the catalog or portions thereof

c) Security model for access to the catalog or portions thereof

1.3 Terminology

Annotation
Free-form text providing human-readable information about a resource or entry.

Catalog
A collection of entries that provides information about a set of resources.

Classifier
A URI attached to a resource to indicate support for the thing identified by the URI such as a feature, specification, etc.

Entry
A portion of a catalog used to describe a resource or group of resources.

EPR
This specification frequently uses EPR as shorthand notation for a WS-Addressing Endpoint Reference.

GED
This specification frequently uses GED as shorthand notation for an XML-Schema global element declaration.

Metadata

Information about a resource that provides additional data about the purpose, limits, capabilities, etc. of the resource that is not necessarily embodied directly within the resource.

MetaEPR
A template mechanism based on the EPR structure defined in WS-Addressing that is used to generate an actual EPR from substitution parameter values.

MetaURI
A template mechanism based on the URI structure used to generate an actual URI from substitution parameter values.

Resource
An entity of interest that can have an XML representation.

URI
Uniform Resource Identifier as defined in [RFC 3986].

1.4 XML Namespaces

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

```
http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog
```

Table 1 lists XML namespaces that are used in this specification. The choice of any namespace prefix is arbitrary and not semantically significant.

Table 1: Prefixes and XML Namespaces used in this specification.

Prefix	XML Namespace	Specification(s)
wsrc	http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog	This document
wsa	http://www.w3.org/2005/08/addressing	[WS-Addressing]
wsa04	http://schemas.xmlsoap.org/ws/2004/08/addressing	[WS-Addressing W3C Submission]
mex	http://schemas.xmlsoap.org/ws/2004/09/mex	[WS-MetadataExchange]
wsdl	http://schemas.xmlsoap.org/wsdl/	[WSDL 1.1]
xs	http://www.w3.org/2001/XMLSchema	[XML Schema]

1.5 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 RFC 2119 [RFC 2119].

This specification uses the following syntax to define outlines for XML elements:

- The syntax appears as an XML instance, but values in italics indicate data types instead of literal values.
- Characters are appended to elements and attributes to indicate cardinality:
  - "?" (0 or 1)
  - "*" (0 or more)
  - "+" (1 or more)

- The character "|" is used to indicate a choice between alternatives.
- The characters "(" and ")" are used to indicate that contained items are to be treated as a group with respect to cardinality or choice.
- The characters "[" and "]" are used to call out references and property names.
- The use of {xs:any} indicates a point of extensibility that allows other child content to be added. An ellipsis (i.e., "...") indicates a point of extensibility that allows other attributes to be added. Additional children and/or attributes MAY be added at the indicated extension points but MUST NOT contradict the semantics of the parent and/or owner, respectively.
- XML namespace prefixes (see Table 1) are used to indicate the namespace of the element being defined.

1.6 Compliance

A document is not compliant with this specification if it fails to satisfy one or more of the MUST, MUST NOT, or REQUIRED statements herein.

Normative text within this specification takes precedence over the XML Schema, which in turn takes precedence over outlines, which in turn take precedence over examples.

2. Catalog Structure

2.1 Introduction

A catalog provides information about a set of management resources. This information allows for classification, linking and discovery of relevant resources. The catalog provides additional information about how to access the resource. This information is included in the catalog to allow for discovery of relevant resources; however the actual resource is the authoritative source of this information.

The outer element is the Catalog element which contains zero or more Entry elements. An Entry element describes a single resource or a group of resources.

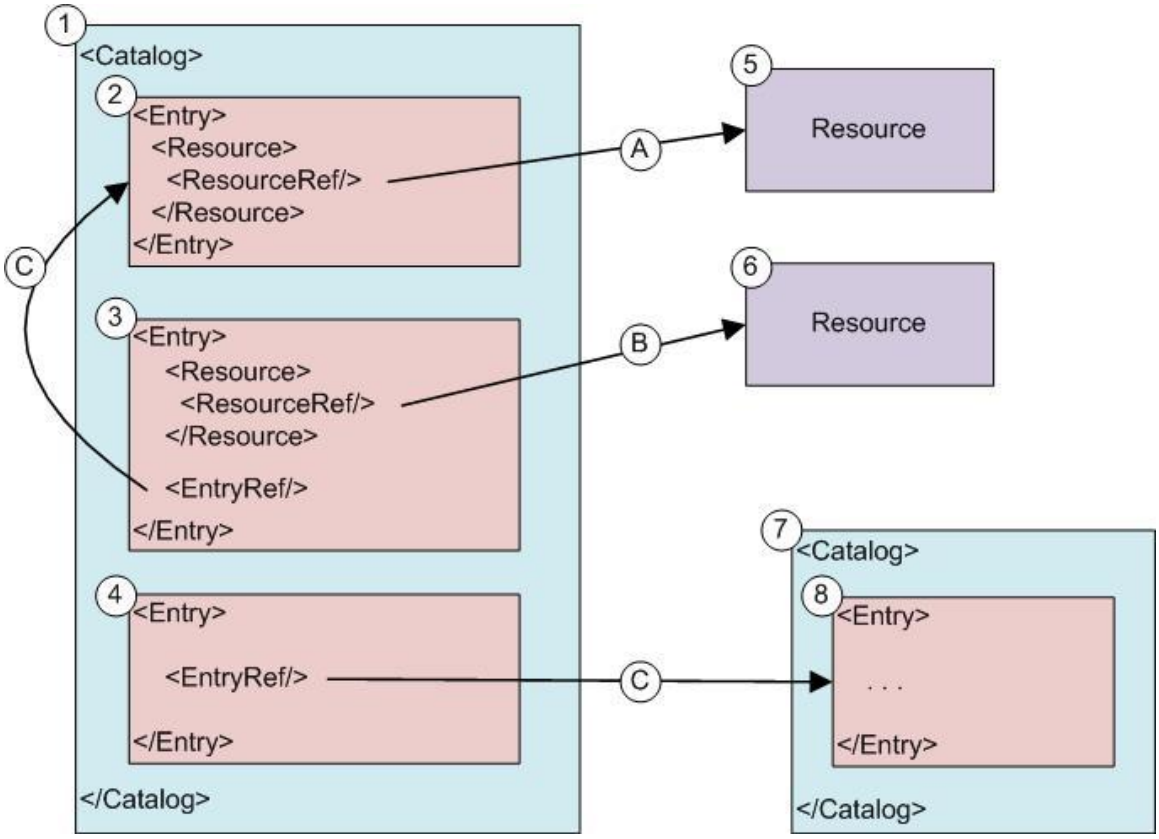
Certain Entry elements represent resources. These entries contain a Resource element that refers to that resource via one or more ResourceRef elements.

Certain Entry elements exist for organizational purposes. These entries can contain multiple EntryRef elements that link to other Entry elements. Graphs of catalog entries such as trees can be described using this technique.

In the figure below, the Catalog (1) contains three Entry elements (2-4). The first Entry element (2) contains a reference (A) to an actual resource (5) using a ResourceRef element. The second Entry element (3) also represents a resource (6) but additionally contains a reference (C) to the first Entry element to express a relationship between the two resources.

Entries can be related to entries in another catalog via a remote EntryRef. The third Entry element (4) is related to an Entry element (8) in another Catalog (7) by way of the remote EntryRef (C). Cross catalog references establish relationships but do not imply the remote entry (8) is a part of the catalog containing the reference (1).

Figure 1



This specification establishes the schema for the catalog and its internal elements. It is beyond the scope of this specification to establish how the catalog or individual entries are accessed by any specific Web service protocol.

2.2 Semantics of Entry

An Entry element can be used to describe an IT resource such as a specific disk. Such an entry would describe that the specific disk exists and how to use Web service protocols to retrieve the resource representation of that disk.

There are many ways to map an entry to an IT resource, especially when the IT resource can have more than one EPR such as access over HTTP or HTTPS. Some of the choices available to the designer include:

a) Model the IT resource as a single Entry element with multiple ResourceRef elements, each containing a different EPR to access the resource.

b) Model the IT resource as several Entry elements, one for each EPR, linked together as peers. (See the alternates in Section 3.5.1.2).

c) Model the IT resource as several Entry elements, one for each EPR and an additional Entry element for a folder, linked together in a directory-like structure. (See hierarchies in Section 3.5.1.1).

The decision criteria for choosing how a resource should be modeled are beyond the scope of this specification. However, it is RECOMMENDED that profiles select a single approach for modeling specific IT resources and that an instance of the catalog uses a single approach.

An entry that is used to describe a specific IT resource SHOULD NOT be overloaded to reference more than one IT resource. For example, an entry describing access to a specific hard disk C: SHOULD NOT also describe access to another specific hard disk D:.

An Entry element can also be used to describe a resource type or a collection of IT resources such as disks. Such an entry would describe that disks exist and how to use Web service protocols that apply to all disks in the collection.

There are many ways to map an entry to a collection of IT resources. Some of the choices available to the designer include:

a) Use N+1 Entry elements in which one Entry element is used to describe the collection itself and the remaining Entry elements are used to describe each of the instances because they are distinct entities from each other. The collection entry can then link to the instance entries.
b) Use one Entry element to represent the collection that includes operations such as iteration, creation, member access, event notification, etc. This assumes that individual instances are not described in the catalog perhaps due to their dynamic nature.
c) Use two Entry elements in which one Entry contains operations on the type such as iteration and creation and another Entry that contains operations on instances of the type such as member access. These entries can then be linked. (This assumes that individual instances are not described in the catalog perhaps due to their dynamic nature.)

The decision criteria for choosing how a collection should be modeled are beyond the scope of this specification. However, it is RECOMMENDED that profiles select a single approach for modeling collections of IT resources and that an instance of the catalog uses a single approach.

An entry that is used to describe a collection of IT resources SHOULD NOT be overloaded to describe a collection of resources of a different type. For example, an entry describing a collection of disks SHOULD NOT also describe a collection of processes.

An Entry element can represent a folder which points to one or more Entry elements for the purposes of structure and grouping.

This specification does not place any constraints on the organization or granularity of entries within a catalog. Entry elements can freely reference other Entry elements and cycles can occur.

2.3 Examples

2.3.1 Introduction

This specification is compatible with a wide variety of data models which require lookup, directory, or catalog services. This section illustrates several brief hypothetical examples of common use cases.

2.3.2 Minimal Catalog of a Single Simple Device Instance

As an example, the following catalog contains a single entry for a logical hard disk. The entry is minimal in that it only contains a single WS-Addressing Endpoint Reference for that resource plus a few annotations and classifiers.

```
(01) <Catalog xmlns="http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog">
(02)  <Entry Id="http://example.com/product/disk002233/disk-c">
(03)   <Annotation xml:lang="en-US">
(04)      This is an example of a disk
(05)    </Annotation>
(06)   <Annotation xml:lang="en-US">hardware</Annotation>
(07)   <Annotation xml:lang="en-US">disk</Annotation>
(08)   <Resource>
(09)     <ResourceRef>
(10)       <ResourceElement
(11)         Namespace="http://schemas.example.com/disk002233.xsd"
(12)         LocalName="LogicalDisk"/>
(13)       <ProtocolAndModelClassifier>
(14)         http://schemas.xmlsoap.org/ws/2004/08/transfer
(15)       </ProtocolAndModelClassifier>
(16)       <ProtocolAndModelClassifier>
(17)         http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer
(18)       </ProtocolAndModelClassifier>
(19)       <Reference xmlns:wsa="http://www.w3.org/2005/08/addressing">
(20)         <wsa:EndpointReference>
(21)           <wsa:Address>
(22)             http://myserver/devices/storage/disk/c
(23)           </wsa:Address>
(24)         </wsa:EndpointReference>
(25)        </Reference>
(26)     </ResourceRef>
(27)   </Resource>
(28)  </Entry>
(29) </Catalog>
```

1. The annotations on lines (06)-(07) indicate that entry relates to "hardware" and "disk". Note that these particular annotations are for example purposes only. They are not defined by this specification.
2. The ResourceElement on lines (10)-(12) indicates that the resource representation has a XML Schema GED whose namespace is **http://schemas.example.com/disk002233.xsd** and whose local name is **LogicalDisk**.

3. The classifiers on line (13)-(18) advertise that the [WS-Transfer] and [WS-ResourceTransfer] protocols are supported for accessing the resource.
4. The EPR on lines (20)-(24) constitute the actual address to be used in a WS-Transfer or WS-ResourceTransfer operation to retrieve the resource.

The entry could be extended with additional annotations, classifiers indicating specific protocol operations ("Get" vs. "Put") from WS-Transfer/WS-ResourceTransfer, the inclusion of the WSDL as metadata (as shown in Appendix I.B), and other useful items.

2.3.3 Cataloging a Class of Resources

In some cases, the catalog entry describes a class of resources, rather than a specific IT resource as in the previous example. If there are a large number of instances of a common class or they are highly dynamic, it might be prohibitively expensive or impossible to keep a catalog containing one Entry element for each instance up-to-date. In such cases, it is often desirable to describe the class as a single Entry element and indicate how an instance can be addressed by an algorithm for generating addresses to the instances. The previous example in 2.3.2 might now appear as:

```
(01) <Entry Id="http://example.com/product/disk002233/disk"
(02)     xmlns="http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog">
(03)   <Annotation xml:lang="en-US">
(04)     This is an example of a disk class
(05)   </Annotation>
(06)   <Annotation xml:lang="en-US">hardware</Annotation>
(07)   <Annotation xml:lang="en-US">disk</Annotation>
(08)   <Resource>
(09)    <ResourceElement
(10)      Namespace="http://schemas.example.com/disk002233.xsd"
(11)      LocalName="LogicalDisk"/>
(12)    <ProtocolAndModelClassifier>
(13)       http://schemas.xmlsoap.org/ws/2004/08/transfer
(14)    </ProtocolAndModelClassifier>
(15)    <ProtocolAndModelClassifier>
(16)       http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer
(17)    </ProtocolAndModelClassifier>
(18)    <ResourceRef>
(19)      <Reference>
(20)        <MetaEPR AddressingVersions=
(21)             "http://schemas.xmlsoap.org/ws/2005/08/addressing">
(22)         <ParameterMap>
(23)           <Parameter Token="DISK" QName="xs:string" QNameType="simpleType">
(24)              <Description xml:lang="en-US">The drive letter.</Description>
(25)              <Example>c</Example>
(26)           </Parameter>
(27)         </ParameterMap>
(28)         <Address> http://myserver/devices/storage/disk/{DISK} </Address>
(29)        </MetaEPR>
```

```
(30)      </Reference>
(31)    </ResourceRef>
(32)  </Resource>
(33) </Entry>
```

This example is the same as the previous one except that the EPR has been replaced with a MetaEPR on lines (20)-(29). The parameter map on lines (22)-(27) indicates that an EPR to a given disk can be constructed given the drive letter by filling it into the address of the EPR as shown on line (28).

2.3.4 Folders and Links

The following example establishes a directory of disk instances, such as the one in 2.3.2:

```
(01) <Entry Id="http://example.com/product/disk002233/disks"
(02)     xmlns="http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog">
(03)   <Classifier>
(04) http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog/classifiers/displayRo
     ot
(05)   </Classifier>
(06)   <Annotation xml:lang="en-US">
(07)     This is an example of a directory of entries
(08)   </Annotation>
(09)   <EntryRef
(10)   Role="http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog/roles/child">
(11)     <EntryId>http://example.com/product/disk002233/disk-c</EntryId>
(12)   </EntryRef>
(13)   <EntryRef
(14)   Role="http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog/roles/child">
(15)     <EntryId>http://example.com/product/disk002233/disk-d</EntryId>
(16)   </EntryRef>
(17) </Entry>
(18)
(19) <Entry Id="http://example.com/product/disk002233/disk-c">
(20)   <Resource>
(21)    <ResourceRef> ... </ResourceRef>
(22)   </Resource>
(23)   <EntryRef
(24)  Role="http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog/roles/parent">
(25)     <EntryId>http://example.com/product/disk002233/disks</EntryId>
(26)   </EntryRef>
(27) </Entry>
(28)
(29) <Entry Id="http://example.com/product/disk002233/disk-d"> ... </Entry>
```

The above example shows an Entry element acting as a folder of disks. Line (04) establishes the entry as the root of a directory-like structure by declaring itself the starting point for navigation. Lines (11) and (15) then contain EntryRef elements which point to entries for each disk.

3. Catalog Elements

This section discusses the XML representation of the catalog structure in detail. The overall structure of the catalog is shown below. The following sections discuss each element in turn.

```
(01) <wsrc:Catalog
(02)    xmlns:wsrc="http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog" ... >
(03)
(04)    <wsrc:Entry Id="xs:anyURI" ...>
(05)       <wsrc:Descriptor ...> wsrc:DescriptorType </wsrc:Descriptor> ?
(06)       <wsrc:Classifier ...> xs:anyURI </wsrc:Classifier> *
(07)       <wsrc:Annotation xml:lang="..."? ...> xs:string </wsrc:Annotation> *
(08)
(09)       <wsrc:Resource ...>
(10)          <wsrc:ResourceRef ...>
(11)              <wsrc:ResourceElement Namespace="xs:anyURI"
(12)                                    LocalName="xs:NCName"/> ?
(13)              <wsrc:ProtocolAndModelClassifier ...>
(14)                  xs:anyURI
(15)              </wsrc:ProtocolAndModelClassifier> *
(16)              <wsrc:Reference ...>
(17)              (  <wsrc:URI> xs:anyURI </wsrc:URI> |
(18)                 <wsrc:MetaURI ...> wsrc:MetaURIType </wsrc:MetaURI> + |
(19)                 <wsrc:MetaEPR ...> wsrc:MetaEPRType </wsrc:MetaEPR> + |
(20)                 <wsa:EndpointReference ...>
(21)                   wsa:EndpointReferenceType
(22)                 </wsa:EndpointReference> |
(23)                 {xs:any} )
(24)              </wsrc:Reference>
(25)              <mex:Metadata ...> ... </mex:Metadata> ?
(26)              {xs:any}*
(27)          </wsrc:ResourceRef> +
(28)          <wsrc:ResourceDiscoveryProperties ...>
(29)              {xs:any} *
(30)          </wsrc:ResourceDiscoveryProperties> ?
(31)          {xs:any}*
(32)       </wsrc:Resource> ?
(33)
(34)       <wsrc:EntryRef Role="xs:anyURI" ...>
(35)          <wsrc:EntryId> xs:anyURI </wsrc:EntryId>
(36)          <wsrc:RemoteRef RefType="Catalog|Entry" ...>
(37)             <wsrc:ProtocolClassifier ...>
(38)                 xs:anyURI
(39)             </wsrc:ProtocolClassifier>*
(40)             <wsrc:Reference ...>
(41)              (  <wsrc:URI> xs:anyURI </wsrc:URI> |
(42)                 <wsa:EndpointReference ...>
(43)                   wsa:EndpointReferenceType
(44)                 </wsa:EndpointReference> |
```

```
(45)                 {xs:any} )
(46)             </wsrc:Reference>
(47)             <mex:Metadata ...> ... </mex:Metadata> ?
(48)             {xs:any}*
(49)          </wsrc:RemoteRef> *
(50)          {xs:any}*
(51)       </wsrc:EntryRef> *
(52)
(53)       {xs:any}*
(54)    </wsrc:Entry> *
(55)
(56)    {xs:any}*
(57) </wsrc:Catalog>
```

3.1 Catalog

A catalog is a document that has a root Catalog element that contains zero or more Entry elements.

The structure of a Catalog element is described below:

```
(01) <wsrc:Catalog
(02)    xmlns:wsrc="http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog" ... >
(03)   <wsrc:Entry ...> wsrc:EntryType </wsrc:Entry> *
(04)   {xs:any}*
(05) </wsrc:Catalog>
```

The following describes additional constraints on the outline listed above:

wsrc:Catalog
: This is the wrapper element which embodies the catalog document.

wsrc:Catalog/wrsc:Entry
: Zero or more elements, each of which contains a logical description of a resource, folder, etc. as described in Section 2.2.

wsrc:Catalog/*{xs:any}*
: This extension point allows other specifications/profiles to add additional non-entry information to the catalog.

A catalog with no Entry elements is legal and can occur in some cases where the catalog reflects a dynamic data source which currently has no available resources.

3.2 Advertising

Entries can advertise information about aspects of the resource they represent. Advertisements can describe 'what' the entry represents or 'how' to talk to the resource. This section provides an overview of how to advertise the information known about a particular resource or type of resource in the catalog.

Advertisements describing 'what' an entry represents are included in the Entry element (see Section 3.3) and include URIs, human-readable descriptions (Annotations) and other structured XML information.

Advertisements describing 'how' to talk to a resource are included in the ResourceRef element (see Section 3.4) and include URIs, schematic types, metadata (such as WS-Policy), and other structured XML information.

When the differentiation of a resource is best achieved by using a property from the resource instance itself or summaries of resource data, such information can be included in the ResourceDiscoveryProperties element (see Section 3.4).

The advertisement mechanisms described in this specification provide a number of choices from high-level to granular. It is RECOMMENDED that advertisements make use of existing mechanisms and utilize specific advertisements with well defined meanings.

It is RECOMMENDED that profiles select one approach from the mechanisms listed above for advertising any particular feature, specification, or identifying property to reduce the possibility of conflicting classification and policy assertions.

3.2.1 Advertising Using URIs

URIs can be used within an entry to advertise that the resource belongs to a well known class in a classification scheme. Classification schemes can be used to advertise capabilities, compatibilities and non-technical classification such as geographic location of a resource or organization owning a resource.

Existing URIs with well defined meanings can be used for advertising. For example, many SOAP specifications define WSA Action URIs that identify specific operations.

Specifications often define URIs that can be used to identify the specification as a whole. These URIs can be used to advertise that the resource implements some part of the specification. This might be a high-level advertisement only and does not indicate support for a particular set of optional features. More granular advertisements can be used to refine the supported features.

An entry can advertise relevant classifiers in a simple list but there is no implication of dependent support between multiple classifiers in an advertisement. In the following example, the set of Classifier URIs advertises that the resource implements WS-ResourceTransfer, WS-Transfer, and the wsa:Action for "Get" from WS-Transfer:

- http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer
- http://schemas.xmlsoap.org/ws/2004/09/transfer
- http://schemas.xmlsoap.org/ws/2004/09/transfer/Get

In this list, the XML namespace defined by WS-ResourceTransfer is used to advertise that the resource implements some portion of WS-ResourceTransfer.

This specification defines no mechanism for grouping or correlating related classifiers. If more rigorous grouping is required, other specifications/profiles can include combinations of classifiers using grouping languages such as [WS-Policy] or Resource Description Framework [RDF].

3.2.2 Advertising Using WS-Policy

When the specifications defining the protocols for connecting to the resource include WS-Policy assertions, those WS-Policy assertions can be used in the advertisement of the resource in a catalog. All available WS-Policy assertions do not have to be advertised, only those useful for selecting Entries or References.

Advertising WS-Policy assertions makes use of the mechanism defined in WS-MetadataExchange for including metadata in EPRs. This allows for the endpoint WSDL (which can have embedded policy) or standalone policy to be included (policy of the Endpoint Policy Subject).

Example of a resource endpoint advertising support for WS-ReliableMessaging:

```
(01) <wsrc:Resource ...>
(02)   <wsrc:ResourceRef ...>
(03)     <wsrc:Reference>
(04)       <wsa:EndpointReference>
(05)         ...
(06)         <wsa:Metadata>
(07)           <mex:Metadata ...>
(08)             <mex:MetadataSection
(09)                    Dialect="http://schemas.xmlsoap.org/ws/2004/09/policy">
(10)               <wsp:Policy>
(11)                 <wsrmp:RMAssertion/>
(12)               </wsp:Policy>
(13)             </mex:MetadataSection>
(14)           </mex:Metadata>
(15)         </wsa:Metadata>
(16)       </wsa:EndpointReference>
(17)     </wsrc:Reference>
(18)   </wsrc:ResourceRef>
(19) </wsrc:Resource>
```

3.3 Entry

An Entry element typically describes a resource, how it can be addressed, what its capabilities are, and other useful information which helps users and tools decide if the resource is of interest. An entry can provide keywords for search engines to determine the relevancy of a particular resource.

The structure of an Entry element is described below:

```
(01) <wsrc:Entry Id="xs:anyURI" ...>
(02)    <wsrc:Descriptor ...> wsrc:DescriptorType </wsrc:Descriptor> ?
(03)    <wsrc:Classifier ...> xs:anyURI </wsrc:Classifier> *
(04)    <wsrc:Annotation xml:lang="..."? ...> xs:string </wsrc:Annotation> *
(05)    <wsrc:Resource ...> wsrc:ResourceType </wsrc:Resource> ?
(06)    <wsrc:EntryRef ...> wsrc:EntryReferenceType </wsrc:EntryRef> *
(07)    {xs:any}*
```

```
(08) </wsrc:Entry>
```

The following describes additional constraints on the outline listed above:

wsrc:Entry
: This is the wrapper which embodies a single catalog entry.

wsrc:Entry/@Id
: This attribute is REQUIRED and contains a URI that MUST uniquely identify the current entry. This identity MUST be globally unique, as entries might be cached and need to be unambiguously identified at some later time or location. The exact ownership of who sets this attribute is beyond the scope of this specification.

wsrc:Entry/wsrc:Descriptor
: This element, if present, contains a description of the resource indicating vendor, etc. See Section 3.3.1.

wsrc:Entry/wsrc:Classifier
: Zero or more elements, each containing a classifier URI which advertises a 'what' aspect of the entry or resource. See Section 3.2.

wsrc:Entry/wsrc:Annotation
: Zero or more elements, each containing a string which describes some human readable aspect of the Entry. Annotations are user-defined notes that are typically scratchpad areas intended primarily for users, deployers, and implementations of catalogs. Annotations SHOULD not be used for formal/structured advertisements.

: An Annotation MAY contain any text and SHOULD be identified by an xml:lang attribute to indicate the language of the text.

wsrc:Entry/wsrc:Resource
: This element, if present, contains information about how to communicate with the resource represented by this entry. If an entry represents a grouping and not an IT resource, then this element will be absent.

wsrc:Entry/wsrc:EntryRef
: Zero or more elements, each of which establishes a relationship with another entry. See Section 3.5 for more information on linking entries.

wsrc:Entry/{*xs:any*}
: This extension point allows additional information about an entry to be included.

Entries use URIs as identifiers rather than xml:id because the identity of the Entry can be used outside of a particular catalog document. For example, an Entry can be referenced from another catalog so the entry id needs to be globally unique.

3.3.1 Descriptor

The Descriptor element allows catalog authors to provide additional information including vendor name, link to additional info, etc. This information is neither advertising of capabilities (such as Classifiers) nor free form text (such as Annotations), but structured information describing the resource.

The structure of a Descriptor block is described below:

```
(01) <wsrc:Descriptor ...>
(02)     <wsrc:DisplayName xml:lang="xs:language"?> xs:string
     </wsrc:DisplayName>*
```

```
(03)     <wsrc:Publisher> xs:string </wsrc:Publisher> ?
(04)     <wsrc:PublisherURL> xs:anyURI </wsrc:PublisherURL> ?
(05)     <wsrc:ResourceURL> xs:anyURI </wsrc:ResourceURL> ?
(06)     <wsrc:Version> xs:string </wsrc:Version> ?
(07)     <wsrc:Created> xs:datetime </wsrc:Created> ?
(08)     <wsrc:Updated> xs:datetime </wsrc:Updated> ?
(09)     {xs:any} *
(10) </wsrc:Descriptor>
```

The following describes additional constraints on the outline listed above:

wsrc:Descriptor
: An element that provides structured information about the entry or resource.

wsrc:Descriptor/wsrc:DisplayName
: This element, if present, contains the name of the entry or resource to be displayed to an end user. This element MAY be repeated in different languages (at most once per language).

  When a name is expressed in a specific language, it SHOULD carry the xml:lang attribute to signify this. When a name does not have an associated language, the xml:lang attribute SHOULD be omitted.

wsrc:Descriptor/wsrc:Publisher
: This element, if present, contains the name of the vendor of the resource.

wsrc:Descriptor/wsrc:PublisherURL
: This element, if present, contains a URL providing more information about the vendor listed in the Publisher element.

wsrc:Descriptor/wsrc:ResourceURL
: This element, if present, contains a URL providing more information about the resource.

wsrc:Descriptor/wsrc:Version
: This element, if present, contains the version of the resource.

wsrc:Descriptor/wsrc:Created
: This element, if present indicates when the catalog entry was created.

wsrc:Descriptor/wsrc:Updated
: This element, if present, indicates when the catalog entry was last updated.

wsrc:Descriptor/{xs:any}
: This extensibility point allows additional descriptive information to be included.

3.4 Resource

A Resource element is used to describe access to the IT resource which the entry represents. The Resource element contains one or more ResourceRef elements each of which indicates how the resource can be reached by a Web service operation.

The ResourceRef element can also capture the XML schema of the resource. However, in some cases, a resource might not have a XML representation in which case the ResourceRef provides a description for interacting with the resource.

The structure of a Resource element is as follows:

```
(01) <wsrc:Resource ...>
```

```
(02)   <wsrc:ResourceRef ...>
(03)     <wsrc:ResourceElement Namespace="xs:anyURI"
(04)                           LocalName="xs:NCName"/> ?
(05)     <wsrc:ProtocolAndModelClassifier ...>
(06)        xs:anyURI
(07)     </wsrc:ProtocolAndModelClassifier> *
(08)     <wsrc:Reference> wsrc:ParameterizableReferenceType </wsrc:Reference>
(09)     <mex:Metadata ...> ..... </mex:Metadata> ?
(10)     {xs:any}*
(11)   </wsrc:ResourceRef> +
(12)   <wsrc:ResourceDiscoveryProperties ...>
(13)      {xs:any}*
(14)   </wsrc:ResourceDiscoveryProperties ...> ?
(15)   {xs:any}*
(16) </wsrc:Resource>
```

The following describes additional constraints on the outline listed above:

wsrc:Resource
: This is the wrapper which contains access information for a resource.

wsrc:Resource/wsrc:ResourceRef
: One or more elements containing any reference information to an endpoint for the resource. If there is more than one reference, this element MAY be repeated.

wsrc:Resource/wsrc:ResourceRef/wsrc:ResourceElement
: This element, if present, is equivalent to the QName used as the root element of the resource representation. The QName has been separated into individual attributes to enhance searching.

wsrc:Resource/wsrc:ResourceRef/wsrc:ResourceElement/@Namespace
: The XML Namespace of the ResourceElement. This is separate from the LocalName attribute to facilitate searching.

wsrc:Resource/wsrc:ResourceRef/wsrc:ResourceElement/@LocalName
: The XML element name of the ResourceElement. This is separate from the Namespace attribute to facilitate searching.

wsrc:Resource/wsrc:ResourceRef/wsrc:ProtocolAndModelClassifier
: Zero or more elements, each containing a classifier URI which advertises a 'how' aspect of the reference to the resource. See Section 3.2.

wsrc:Resource/wsrc:ResourceRef/wsrc:Reference
: This contains the actual reference in the form of a URI, EPR, etc as described in Section 3.4.1.

wsrc:Resource/wsrc:ResourceRef/mex:Metadata
: This element, if present, contains metadata relating to the remote resource. It MAY contain a subset, superset, or cached copy of the metadata which might be accessible via WS-MetadataExchange at the resource endpoint, but can also contain other metadata relating to the use or policies about the resource. If present, it MUST be the first element to make use of the xs:any extensibility point.

wsrc:Resource/wsrc:ResourceRef/{*xs:any*}
: This extensibility point allows additional information about the ResourceRef to be included.

wsrc:Resource/wsrc:ResourceDiscoveryProperties

This element, if present, contains information about the resource for discovery purposes. Information about resources SHOULD be limited to stable data. For example, on an active mounted disk the drive letter might be a stable property that could be mapped as an element to be used for discovery whereas free space is a volatile property and SHOULD be retrieved from the disk resource directly.

wsrc:Resource/{*xs:any*}
This extensibility point allows additional information about the resource to be included.

While a Resource element can include multiple ResourceRef elements, they MUST all reference the same IT resource but MAY differ by operation or address. A Resource element SHOULD NOT contain separate ResourceRef elements for two different resources (e.g. Disk A: and Disk B:); instead two separate Entry elements SHOULD be used to describe these two separate instances of Disk.

3.4.1 Reference

A reference to an IT resource can take different forms based upon its addressing technique. The catalog might contain an actual address or the reference might require additional information before an actual address can be used.

The structure of a Resource/ResourceRef/Reference element is as follows:

```
(17) <wsrc:Reference ...>
(18)   ( <wsrc:URI ...> xs:anyURI </wsrc:URI> |
(19)     <wsrc:MetaURI ...> wsrc:MetaURIType </wsrc:MetaURI> + |
(20)     <wsrc:MetaEPR ...> wsrc:MetaEPRType </wsrc:MetaEPR> + |
(21)     <wsa:EndpointReference>
(22)         wsa:EndpointReferenceType
(23)     </wsa:EndpointReference> |
(24)     {xs:any} )
(25) </wsrc:Reference>
```

The following describes additional constraints on the outline listed above:

wsrc:Reference
This is the wrapper for a choice of reference types as listed below.

wsrc:Reference/wsrc:URI
This element, if present, indicates that the resource is identified by the given URI.

wsrc:Reference/wsrc:MetaURI
This element, if present, indicates that the resource can be identified by a URI once parameter values are substituted into the MetaURI. See Section 3.6 for information on the use of meta references to generate a working URI from a MetaURI.

wsrc:Reference/wsrc:MetaEPR
This element, if present, indicates that the resource can be reached by an EPR once parameter values are substituted into the MetaEPR. See Section 3.6 for information on the use of meta references to generate a working EPR from a MetaEPR.

wsrc:Reference/wsa:EndpointReference
This element, if present, indicates that the resource is accessed via the specified EPR. Classifiers can indicate which Web service operations are applicable.

wsrc:Reference/{*xs:any*}

This is an extensibility point to capture other addressing models not listed above such as the WS-Addressing W3C submission version.

Multiple MetaEPRs or MetaURIs within the same ResourceRef SHOULD represent the same semantic use of the resource and only differ by their set of parameters.

3.5 EntryRef

An EntryRef element is used to establish a link from the current entry to another entry. For example, a link can describe a logical successor or a predecessor. The EntryRef element MAY be repeated as many times as is necessary to establish all the required relationships to other entries.

The structure of an EntryRef element is as follows:

```
(01) <wsrc:EntryRef Role="xs:anyURI" ...>
(02)     <wsrc:EntryId> xs:anyURI </wsrc:EntryId>
(03)     <wsrc:RemoteRef RefType="Catalog|Entry" ...>
(04)         <wsrc:ProtocolClassifier ...>
(05)             xs:anyURI
(06)         </wsrc:ProtocolClassifier> ?
(07)         <wsrc:Reference ...> wsrc:ReferenceType </wsrc:Reference>
(08)         <mex:Metadata ...> ... </mex:Metadata> ?
(09)         {xs:any} *
(10)     </wsrc:RemoteRef> *
(11)     {xs:any} *
(12) </wsrc:EntryRef>
```

The following describes additional constraints on the outline listed above:

wsrc:EntryRef

This is the wrapper which embodies the EntryRef.

wsrc:EntryRef/@Role

A REQUIRED URI indicating the Role the referenced Entry plays with respect to the current Entry. See Section 3.5.1 for more information about roles.

wsrc:EntryRef/wsrc:EntryId

This REQUIRED element indicates the Id of the Entry being referenced.

wsrc:EntryRef/wsrc:RemoteRef

Zero or more elements used primarily when the Entry is not in the current catalog, but is a remote reference. If present, all RemoteRef elements MUST refer to the same Entry, but can differ by EPR, supported operations, etc.

wsrc:EntryRef/wsrc:RemoteRef/@RefType

A REQUIRED attribute which indicates whether the following reference is to a Catalog or an Entry resource and MUST be one of the values "Catalog" or "Entry". The EntryRef always refers to a specific entry as the wsrc:EntryId is required; however, the website or Web service that publishes the Entry being referenced might only provide access to the Catalog as a whole. In this case, the EntryRef is resolved by accessing the Catalog document and then selecting the Entry identified by the appropriate id.

wsrc:EntryRef/wsrc:RemoteRef/wsrc:ProtocolClassifier

Zero or more elements, each containing a classifier URI which advertises a 'how' aspect of the reference to the entry or catalog. See Section 3.2.

wsrc:EntryRef/wsrc:RemoteRef/wsrc:Reference
This contains the actual reference in the form of a URI, EPR, etc as described in Section 3.5.2.

wsrc:EntryRef/wsrc:RemoteRef/mex:Metadata
This element, if present, contains metadata relating to the remote entry or catalog. It MAY contain a subset, superset, or cached copy of the metadata which might be present on the entry or catalog, but can also contain other metadata relating to the use or policies about the entry or catalog. If present, it MUST be the first element to make use of the xs:any extensibility point.

wsrc:EntryRef/wsrc:RemoteRef/{*xs:any*}
This extensibility point allows additional information about the RemoteRef to be included.

wsrc:EntryRef/{*xs:any*}
This extensibility point allows additional information about the EntryRef to be included.

References to entries in other catalogs capture the relationship between the two entries, but do not extend the current catalog by including the referenced entry. Implementations MAY provide mechanisms to query EntryRef relationships and traverse links, but the resulting entries might be in different catalogs.

3.5.1 Roles

Roles are identified using a URI and indicate the relationship between the entries.

Note that the links between entries are not necessarily a tree and that graphs can be described which contain cycles. This specification places no predefined limits on what can be defined as a role.

This specification defines a number of common roles. Other specifications/profiles can define additional roles identified with their own URIs.

3.5.1.1 Hierarchies

This specification defines two roles that can be used to organize resources into trees and folders:

- http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog/roles/parent
  - indicates that the referenced entry is a parent folder
- http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog/roles/child
  - indicates that the referenced entry is a child of the current folder

When entries are organized into trees, a client needs to know which entries are the starting points of the tree. The following classifier URI indicates that the entry can be used as the starting point for displaying a tree. A catalog can contain more than one entry labeled as a starting point. A client can start with entries containing this classifier and follow EntryRef links to other entries in the catalog. This classifier is identified by the following URI:

http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog/classifiers/displayRoot

The following example describes a folder with two sub trees:

```
(01) <Catalog xmlns="http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog">
(02)     <Entry Id="Top">
(03)         ...
(04)         <Classifier>http://.../classifiers/displayRoot</Classifier>
(05)         <EntryRef Role="http://.../roles/child">
(06)             <EntryId>SubTreeA</EntryId>
(07)         </EntryRef>
(08)         <EntryRef Role="http://.../roles/child">
(09)             <EntryId>SubTreeB</EntryId>
(10)         </EntryRef>
(11)     </Entry>
(12)     <Entry Id="SubTreeA"> ... </Entry>
(13)     <Entry Id="SubTreeB"> ... </Entry>
(14) </Catalog>
```

In this example, the displayRoot classifier on line (04) indicates that the hierarchy starts with the Top entry.

The entry Top in turn has two branches for the hierarchy into two new catalog Entry elements, one for SubTreeA and one for SubTreeB.

Note in this example that each EntryRef uses the "Child" role. However, if each referenced entry also needs a pointer back to the parent, then an additional EntryRef element would be added in each SubTree entry. This EntryRef element contains the EntryId of the parent entry and uses the "Parent" role. For example, the following shows entry SubTreeB pointing back to its parent:

```
(15) <Catalog xmlns="http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog">
(16)     <Entry Id="Top"> ... </Entry>
(17)     <Entry Id="SubTreeA"> ... </Entry>
(18)     <Entry Id="SubTreeB">
(19)         ...
(20)         <EntryRef Role="http://.../roles/parent">
(21)             <EntryId>Top</EntryId>
(22)         </EntryRef>
(23)     </Entry>
(24) </Catalog>
```

3.5.1.2 Alternates

The "Alternate" role is used to link two Entries that could have been the same Entry but are separated for some reason such as security, etc. This role is identified by the following URI:

http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog/roles/alternate

The alternate role is not transitive so separate links need to be provided for all alternates. Similarly, the alternate role is not symmetricso separate links need to be

provided for both directions. This allows the client to easily navigate without the need for multiple queries.

3.5.2 Reference

References to other entries can take the form of a URI or EPR or other addressing mechanism.

The structure of an EntryRef/RemoteRef/Reference element is as follows:

```
(01) <wsrc:Reference ...>
(02) (  <wsrc:URI> xs:anyURI </wsrc:URI> |
(03)    <wsa:EndpointReference ...>
(04)      wsa:EndpointReferenceType
(05)    </wsa:EndpointReference> |
(06)    {xs:any}
(07) )
(08) </wsrc:Reference>
```

The following describes additional constraints on the outline listed above:

wsrc:Reference
: This is the wrapper for a choice of reference types as listed below.

wsrc:Reference/wsrc:URI
: The element, if present, indicates that the target Entry or Catalog is identified by the given URI.

wsrc:Reference/wsa:EndpointReference
: This element, if present, indicates that the target Entry or Catalog resides at the specific WS-Addressing EPR and might be accessed using techniques such as WS-ResourceTransfer, WS-Transfer, [WS-Enumeration], etc. This specification does not mandate what access mechanism(s) are supported.

wsrc:Reference/{xs:any}
: This is an extensibility point to capture other addressing models not listed above such as the WS-Addressing W3C submission version.

3.6 Meta References

Some references to resources cannot be completely specified by the catalog. For example, a catalog on a website might not know the actual server name on which the resource is to be found. Meta references allow a catalog entry to describe a reference as a combination of a template and parameters that are needed to create a valid reference.

The MetaEPR and MetaURI elements describe how to parameterize an EPR and a URI respectively. These elements MAY occur in a ResourceRef. If multiple MetaEPRs (MetaURIs) exist within the same reference, they MUST refer to the same resource but allow for a different set of substitution parameters. The user of the catalog can choose one of the MetaEPRs (MetaURIs) based upon which set of parameters it knows about.

3.6.1 ParameterMap

The parameters for a meta reference are described in a ParameterMap that includes type information, description and examples.

The following outline describes the structure of the ParameterMap element:

```
(01) <wsrc:ParameterMap ...>
(02)   <wsrc:Parameter Token="xs:NCName" QName="xs:QName"
(03)       QNameType="simpleType | innerValueOfGED | outerValueOfGED" ...>
(04)     <wsrc:Description xml:lang="xs:language"?>xs:string</wsrc:Description>*
(05)     <wsrc:Example> {xs:any;mixed}* </wsrc:Example> *
(06)     {xs:any}*
(07)   </wsrc:Parameter> +
(08) </wsrc:ParameterMap>
```

The following describes additional constraints on the outline listed above:

wsrc:ParameterMap
: This element provides a list of the unresolved tokens.

wsrc:ParameterMap/wsrc:Parameter
: This element describes a single unresolved token.

wsrc:ParameterMap/wsrc:Parameter/@wsrc:Token
: This REQUIRED attribute is the token found in the parameterized elements. Tokens are case sensitive.

wsrc:ParameterMap/wsrc:Parameter/@wsrc:QNameType
: This REQUIRED attribute indicates how to construct the value to be substituted for the token. It MUST be one of three possible values:

  - simpleType – the value to be substituted MUST conform to the simple type identified by the QName attribute.
  - innerValueOfGED – the value to be substituted MUST be the contents of an XML element with the GED given by the QName attribute after removing the GED element.
  - outerValueOfGED – the value to be substituted MUST be the contents of an XML element with the GED given by the QName attribute including the GED element.

wsrc:ParameterMap/wsrc:Parameter/@wsrc:QName
: This REQUIRED attribute identifies either a simple type or a GED that indicates how to construct the value to be substituted for the token.

wsrc:ParameterMap/wsrc:Parameter/wsrc:Description
: The element, if present, provides a human readable description of the purpose of the token. This element MAY be repeated for different languages.

  When a description is expressed in a specific language, it SHOULD carry the xml:lang attribute to signify this. When a description does not have an associated language, the xml:lang attribute SHOULD be omitted.

wsrc:ParameterMap/wsrc:Parameter/wsrc:Example
: This element, if present, provides an example value for the token to demonstrate the syntax if needed. This element MAY be repeated to showcase different syntax structures.

wsrc:ParameterMap/wsrc:Parameter/{*xs:any*}
: This extension point allows additional information about the parameter to be included in the map.

A ParameterMap MUST NOT be used to construct a reference unless values for all parameters are known.

3.6.2 Substitution

A meta reference is a string that contains one or more of the parameters from the ParameterMap. References to parameters are enclosed within curly brace characters. An actual reference is constructed by replacing the brace-enclosed parameters with their actual values.

The following example includes three parameters "a", "b", and "c" that occur within the text surrounded by braces:

```
The sum of {a} and {b} is {c}.
```

Given values of "1", "2" and "3", respectively, this meta-string can be converted into an actual string by replacing the values to yield:

```
The sum of 1 and 2 is 3.
```

There is no limit to the number of substitutions that might be required on a meta string to transform it into an actual string. The processor MUST continue to process brace-surrounded tokens until none remain. The iteration is NOT recursive. If the substituted value itself contains braces, they are not reevaluated as part of the mechanism.

Once substitution is complete, the resulting string needs to be processed and placed into the appropriately typed field in the actual reference. For example, if the resulting string is used within the reference as an xs:anyURI, any leading whitespace needs to be removed.

A ParameterMap MUST NOT be used to construct a reference if it contains a brace-surrounded token that is not declared in the ParameterMap.

In cases where an open brace '{' character is required literally as part of a string, it can appear twice and MUST be replaced by a single open brace character during substitution. The close brace '}' character does not need to be escaped as it only has special meaning when a non-escaped open brace '{' character has been previously encountered and not offset by a close brace '}' character. Once a non-escaped open brace character is encountered, characters are processed until the matching close brace '}' character is found and the enclosed token is replaced by its value in the ParameterMap.

For example, the following meta-string contains a single parameter "Name" and the "{{Hello" sequence in the meta-string would be rendered to simple "{Hello" by the processor.

```
The greeting was "{{Hello}, {Name}".
```

Given a value of "Fred" for Name, this becomes this:

```
The greeting was "{Hello}, Fred".
```

3.6.3 Pre-Defined Parameters

The specification defines two GEDs for use as parameters for MetaEPRs and MetaURIs. This allows programmatic substitution of these parameters without asking the user for their values for every reference in the same catalog.

The following GEDs capture common portions related to addressing:

- wsrc:Host (*xs:string*)
  - Definition: The host name or IP address of the endpoint on which the resource can be found.
  - Example: "example.com" or "123.123.123.0"
- wsrc:Port (*xs:positiveInteger*)
  - Definition: The port number of the endpoint on which the resource can be found.
  - Example: "80" or "443"

Catalog authors SHOULD use these GEDs when the parameterized value has the same meaning as defined for the GED. Catalog authors MAY include the above descriptions and examples as wsrc:Description and wsrc:Example elements for the parameters in additional languages or to provide examples relevant to their domain. Catalog clients can build in default descriptions and examples for these GEDs in case none are provided in the catalog document.

3.6.4 MetaEPR

This element is used when a complete EPR is not available for a resource and the client needs to provide additional information before accessing the resource.

The MetaEPR is structurally similar to an EPR, but contains unresolved tokens which need to be replaced with actual values before use. The client follows a canonical algorithm for examining elements and tokens in the MetaEPR and builds an EPR as the output. The unresolved tokens are described in a ParameterMap that describes the purpose of the token and its type.

The following outline describes the structure of the MetaEPR:

```
(09) <wsrc:MetaEPR AddressingVersions="list of xs:anyURI" ...>
(10)     <wsrc:ParameterMap ...> wsrc:ParameterMapType </wsrc:ParameterMap>
(11)     <wsrc:Address> xs:string </wsrc:Address>
(12)     <wsrc:ReferenceParameters ...> wsrc:MetaEndpointElementType
    </wsrc:ReferenceParameters> ?
(13)     <wsrc:Metadata ...> wsrc:MetaEndpointElementType </wsrc:Metadata> ?
(14)     <wsrc:Any> {xs:any}* </wsrc:Any> ?
(15)     {xs:any}*
(16) </wsrc:MetaEPR>
```

The following describes additional constraints on the outline listed above:

wsrc:MetaEPR

This element is the wrapper and is the analog to the wsa:EndpointReference wrapper.

wsrc:MetaEPR/@AddressingVersions
This REQUIRED list of URIs indicates what versions of WS-Addressing can be used. The generated EPR MUST use one of the possible values in this list as the XML Namespace URI for the root QName and related child elements.

wsrc:MetaEPR/wsrc:ParameterMap
This element defines the parameters to use in constructing the EPR.

wsrc:MetaEPR/wsrc:Address
This element is the analog to the wsa:Address element but can be parameterized with one or more tokens from the ParameterMap.

wsrc:MetaEPR/wsrc:ReferenceParameters
This element, if present, is the analog to the wsa:ReferenceParameters element but can be parameterized with one or more tokens from the ParameterMap.

wsrc:MetaEPR/wsrc:Metadata
This element, if present, is the analog to the wsa:Metadata element but can be parameterized with one or more tokens from the ParameterMap.

wsrc:MetaEPR/wsrc:Any
This element, if present, is the analog to the extensibility point in the EPR but can be parameterized with one or more tokens from the ParameterMap.

wsrc:MetaEPR/{*xs:any*}
This extension point allows additional information about the MetaEPR to be included. Information in this extension point MUST NOT be placed into the computed EPR.

These elements (excluding ParameterMap) have the same meaning as they do in the wsa:EndpointReference, except that string processing rules set above need to be followed before the element can be used in an EPR. Actual values for each parameter MUST be substituted into each string and then an EPR constructed using those strings. E.g. substitute parameters into the value of wsrc:Address and then use the result as the value of the wsa:Address. For wsrc:Any, parameters are substituted and the resulting content is included in the extensibility point of the EPR.

An unprocessed MetaEPR element looks like a normal wsa:EndpointReference, except that certain tokens appear within the body surrounded by curly brace characters:

```
(01) <wsrc:MetaEPR xmlns:map="schema.example"
(02)      AddressingVersions="http://schemas.xmlsoap.org/ws/2005/08/addressing">
(03)    <wsrc:ParameterMap>
(04)        <wsrc:Parameter Token="server" QName="wsrc:Host"
(05)           QNameType="innerValueOfGED"/>
(06)    </wsrc:ParameterMap>
(07)    <wsrc:Address> http://{server}/myService </wsrc:Address>
(08) </wsrc:MetaEPR>
```

In the example above, the URL forming the wsa:Address is parameterized with a token "server" surrounded by braces. The token surrounded by braces indicates that this portion of the address is not known to the catalog author and that the token MUST be resolved to its true value and substituted at the specified location in order to obtain a working EPR.

For example, the above MetaEPR could be processed and the "server" token replaced with an actual IP address to construct the following EPR:

```
(01) <wsa:EndpointReference>
(02)     <wsa:Address> http://192.168.1.191/myService </wsa:Address>
(03) </wsa:EndpointReference>
```

Upon completion of this processing, the EPR is now ready to use as a normal EPR to retrieve a resource.

3.6.4.1 WS-Addressing W3C Submission Version EPRs

A MetaEPR can also represent a wsa04:EndpointReference but the conversion is slightly different than for generating a wsa:EndpointReference. If the @AdressingVersions attribute contains the wsa04 XML namespace, the EPR is generated as above with the exception of the wsrc:Metadata element.

When creating a wsa:EndpointReference, the wsrc:Metadata element is mapped to the wsa:Metadata element. However, wsa04:EndpointReferences do not have a Metadata element, so the contents of the wsrc:Metadata and wsrc:Any MUST both be mapped to the open content portion of the EPR.

For example, the following MetaEPR describes a wsa04:EndpointReference:

```
(01) <wsrc:MetaEPR xmlns:v="http://example.com/addressing"
(02)      AddressingVersions="http://schemas.xmlsoap.org/ws/2004/08/addressing">
(03)   <wsrc:ParameterMap>
(04)       <wsrc:Parameter Token="server" QName="wsrc:Host"
(05)           QNameType="innerValueOfGED"/>
(06)   </wsrc:ParameterMap>
(07)   <wsrc:Address> http://{server}/myService </wsrc:Address>
(08)   <wsrc:ReferenceParameters>
(09)       <v:VendorA> xyz </v:VendorA>
(10)   </wsrc:ReferenceParameters>
(11)   <wsrc:Metadata>
(12)       <v:VendorB> abc </v:VendorB>
(13)   </wsrc:Metadata>
(14)   <wsrc:Any>
(15)       <v:VendorC> 123 </v:VendorC>
(16)   </wsrc:Any>
(17) </wsrc:MetaEPR>
```

The above MetaEPR can be processed and the "server" token replaced with an actual IP address to construct the following WS-Addressing W3C submission version EPR:

```
(01) <wsa04:EndpointReference xmlns:v="http://example.com/addressing">
(02)     <wsa04:Address> http://192.168.1.191/myService </wsa04:Address>
(03)     <wsa04:ReferenceParameters>
(04)         <v:VendorA> xyz </v:VendorA>
(05)     </wsa04:ReferenceParameters>
(06)     <v:VendorB> abc </v:VendorB>
(07)     <v:VendorC> 123 </v:VendorC>
```

```
(08) </wsa04:EndpointReference>
```

Line (06) contains the contents of the wsrc:Metadata element because wsa04:EndpointRefrence has no explicit Metadata element (unlike wsa:EndpointReference).

If the metadata has different forms in the two different versions of the EPR, then two ResourceRef elements MUST be used.

3.6.5 MetaURI

The MetaURI element provides the ability to reference items by a parameterized URI in the same way MetaEPR provides for parameterized EPRs.

The following outline describes the structure of the MetaURI:

```
(01) <wsrc:MetaURI ...>
(02)     <wsrc:ParameterMap ...> wsrc:ParameterMapType </wsrc:ParameterMap>
(03)     <wsrc:TemplateURI> xs:string </wsrc:TemplateURI>
(04)     {xs:any} *
(05) </wsrc:MetaURI>
```

The following describes additional constraints on the outline listed above:

wsrc:MetaURI
: This element defines a parameterized URI.

wsrc:MetaURI/wsrc:ParameterMap
: This element defines the parameters to use in constructing the URI.

wsrc:MetaURI/wsrc:TemplateURI
: This element is a URI parameterized with the tokens from the ParameterMap.

wsrc:MetaURI/{*xs:any*}
: This extension point allows additional information about the MetaURI to be included. Information in this extension point MUST NOT be placed into the computed URI.

The substitution of parameters in the URI follows the same algorithm as used for EPRs. This effectively matches the proposed template model for URIs described by [URI Template].

4. Catalog Access

While this specification only defines the schema of the catalog and its internal elements, this non-normative section outlines types of catalog access and considerations for profiles that define protocols to access the catalog.

4.1 Catalog Types

The catalog has been designed so that it is compatible with several different access paradigms:

a) Some implementations might treat the catalog as a complete XML document and transfer it as a whole. This document might be accessed in a variety of ways:
    a. Retrieved via HTTP from the managed system

b. Retrieved via resource access specifications such as WS-Transfer or WS-ResourceTransfer from the managed system
c. Stored in the local file system and retrieved as a whole using conventional file access mechanisms
d. Retrieved via HTTP from the vendor's website

b) Some implementations might treat the catalog as a collection of entries. The entries might be accessed using a variety of mechanisms:
a. Iteration via specifications such as WS-Enumeration
b. Queried via WS-Enumeration using either XPath filters or more complex join queries across multiple entries.
c. Retrieved via WS-Transfer/WS-ResourceTransfer

c) Some implementations might use a mix of document-based and collection representations and support navigation between them.

It is beyond the scope of this specification to establish how the catalog or individual entries are to be accessed by any specific Web service protocol. Profiles which define specific access models using specific protocols can address the following:

a) Define whether the catalog as a whole can be retrieved as a single document
b) Define whether individual Entry elements can be retrieved by iteration, query, or by a direct "get" of the specific Entry element using some addressing technique
c) Define what types of filter or query dialects are supported for searching, and if any catalog-specific helper dialects are defined.
d) Security and access control of catalog data.

4.1.1 Internet Published Catalogs

Managed systems include small hardware devices with limited on-board storage capacity. These systems might want to publish their catalogs on a website.

Some considerations for designing catalogs for these devices:

- Most ResourceRefs will use MetaEPRs to allow the catalog to be independent of the particular local network where the device is operating. In many cases this will use the wsrc:Host or wsrc:Port as parameters to the URI or Address of the resource with a wsrc:QNameType of "innerValueOfGED" as mentioned in Section 3.6.3.

4.1.2 Database Backed Catalogs

Large systems with variable components might have too many resources to be represented in a single catalog document. These systems can instead generate portions of the catalog on demand from a backing store such as a data base.

Some considerations for designing these catalogs:

- Collections of instances of a common type can use a single ResourceRef with a MetaEPR containing parameters for the instance identities.
- Support for catalog queries that can be easily translated into the native query mechanism of the backing store.
- Don't re-use Entry ids as entries are added and removed since this will defeat the ability of clients to cache subsets of the catalog.

5. Security Considerations

This section describes the security considerations that service providers, requestors, catalog authors, and implementers using catalog information need to consider when providing, consuming and designing a catalog implementation.

Conformance to this specification does not require the recipient of a message or document with catalog information to process any of the WS-ResourceCatalog constructs within if the receiver is not satisfied that the document or message is safe to process.

It is recommended that access to the catalog be secured using mechanisms described in WS-Security or transport-level security such as HTTPS. It is recommended that Catalog documents not be accepted unless they have been received over a secure channel and the integrity of the catalog has been verified or the client has a mechanism to ensure the authenticity and integrity of the source. The mechanisms to establish integrity and secure channels are not defined in this specification and implementations of the catalog should establish appropriate mechanisms to secure the access to catalog contents and provide integrity mechanisms.

The catalog data model also provides no normative means for validating the integrity of individual Entry elements. Catalogs that comprise data from multiple sources will need to define additional mechanisms to secure the contents of the catalog.

5.1 Information Disclosure Threats

A catalog entry is used to represent the capabilities and requirements of a resource and can contain properties of resources and hence might include sensitive information. Malicious consumers can acquire sensitive information and infer service vulnerabilities via the catalog. These threats can be mitigated by requiring authentication and securing access to the catalog or by omitting sensitive information from the catalog. For securing access to the catalog, catalog providers can use transport level mechanisms or mechanisms from other Web Services specifications such as WS-Security [WS-Security].

It is also important to note that the resolution of references in a catalog that require a connection to another resource can result in information disclosure. Information subject to disclosure includes any parameters used in a MetaEPR as well as other information used in creating the request. A consumer of catalog information should establish criteria to mitigate any threats associated with use of references found in a catalog.

5.2 Spoofing and Tampering Threats

If a catalog document is not received over a secure channel with appropriate integrity mechanisms it could be easily tampered with or replaced. It is recommended that catalog documents not be accepted unless the integrity of the

catalog has been verified. Requestors should also check that the source or sources of the catalog, as determined using the integrity mechanism, is actually authorized to provide the information in the catalog document including the entry elements within the catalog.

5.3 Denial of Service Threats and General XML Considerations

Malicious providers might provide a catalog document with a large number of Entry elements, connection alternatives or complex graphs of entries (this is similar to the well-known DTD entity expansion attack). Consumers of a catalog need to anticipate this threat and use an algorithm to limit the resolution of catalog contents with defaults on handling the depth of referencing, depth and nesting of XML content and number of elements in unbounded sequences.

6. Acknowledgements

This specification has been developed as a result of joint work with many individuals and teams, including: Chris Ferris (IBM), Ian Robinson (IBM), Jacob Eisinger (IBM), James Martin (Intel Corporation), John Colgrave (IBM), Kirill Gavrylyuk (Microsoft), Mark Johnson (IBM), Maryann Hondo (IBM), Simeon Pinder (HP), Tony Nadalin (IBM), Tony Storey (IBM), Vince Brunssen (IBM).

7. References

[RDF]
D. Becket, et al, "Resource Description Framework," W3C, February 2004. (See http://www.w3.org/RDF/.)

[RFC 2119]
S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Harvard University, March 1997. (See http://www.ietf.org/rfc/rfc2119.txt.)

[RFC 3986]
T. Berners-Lee, et al, "Uniform Resource Identifier (URI): Generic Syntax," RFC 3986, W3C/MIT, January 2005. (See http://www.ietf.org/rfc/rfc3986.txt)

[URI Template]
J. Gregorio, et al, "URI Template," Oct 2006. (See http://www.ietf.org/internet-drafts/draft-gregorio-uritemplate-00.txt.)

[WS-Addressing]
W3C Recommendation, "Web Services Addressing 1.0 (WS-Addressing)," May 2006. (See http://www.w3.org/2005/08/addressing/.)

[WS-Addressing W3C Submission]
D. Box et al, "Web Services Addressing (WS-Addressing)," August 2004. (See http://www.w3.org/Submission/2004/SUBM-ws-addressing-20040810/.)

[WSDL 1.1]
E. Christensen, et al, "Web Services Description Language 1.1 (WSDL)," March 2001. (See http://www.w3.org/TR/2001/NOTE-wsdl-20010315.)

[WS-Enumeration]
J. Alexander, et al, "Web Services Enumeration (WS-Enumeration)," March 2006. (See http://www.w3.org/Submission/WS-Enumeration/.)

[WS-MetadataExchange]

K. Ballinger, et al, "Web Services Metadata Exchange 1.1 (WS-MetadataExchange)," August 2006. (See http://schemas.xmlsoap.org/ws/2004/09/mex.)

[WS-Policy]
A. Vedamuthu, et al, "Web Services Policy 1.5 - Framework Transfer (WS-Policy)," March 2007. (See http://www.w3.org/TR/ws-policy/)

[WS-ResourceTransfer]
B. Reistad, et al, "Web Services Resource Transfer (WS-ResourceTransfer)," August 2006. (See http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer)

[WS-Security]
OASIS standard, "Web Services Security: SOAP Message Security 1.0" (See http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)

[WS-Transfer]
J. Alexander, et al, "Web Services Transfer (WS-Transfer)," March 2006. (See http://www.w3.org/Submission/WS-Transfer/.)

[XML Schema, Part 1]
H. Thompson, et al, "XML Schema Part 1: Structures," October 2004. (See http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/.)

[XML Schema, Part 2]
P. Biron, et al, "XML Schema Part 2: Datatypes," October 2004. (See http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/.)

Appendix I – Examples

I.A Device Catalog

This catalog models a hardware device that provides management functionality for a computer. It has three components for managing the computer system, sensors, and log records. For example a sensor can monitor the CPU temperature and if a threshold is exceeded it can record a message in the log.

This catalog contains three Entry elements:

- An Entry for the computer system allowing for power up and power down.
- An Entry for the collection of sensors on the device that can monitor the CPU and other things.
- An Entry for the event log that records messages when a sensor threshold is exceeded.

This catalog is intended to be published on the vendor's website, so it uses MetaEPRs to parameterize EPRs with the hostname of the actual device. Since two versions of WS-Addressing are supported, the MetaEPRs contain the relevant WS-Addressing versions.

I.A.1 ComputerSystem Entry

The first Entry element represents the computer system managed by the device. The current state of the system can be retrieved via WS-Transfer/WS-ResourceTransfer and custom actions provide a way to power the system on or off. The entry uses classifiers to advertise supported features and annotations to capture some keywords.

```
(01) <Entry Id="http://example.com/product/xyzdevice/v1.0.2/catalog.xml#cpu">
(02)   <Descriptor>
(03)     <DisplayName xml:lang="en-US"> ComputerSystem </DisplayName>
(04)   </Descriptor>
(05)   <Classifier> http://example.com/classifiers/hardware </Classifier>
(06)   <Classifier> http://example.com/classifiers/powerMgmt </Classifier>
(07)   <Classifier>
(08) http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog/classifiers/displayRo
     ot
(09)   </Classifier>
(10)   <Annotation xml:lang="en-US"> ComputerSystem  </Annotation>
(11)   <Annotation xml:lang="en-US"> Reboot </Annotation>
(12)   <Annotation xml:lang="en-US"> Power </Annotation>
(13)   <Resource>
(14)     <ResourceRef>
(15)       <ResourceElement
(16)          Namespace="http://example.com/product/xyzdevice/v1.0.2/device.xsd"
(17)          LocalName="ComputerSystem"/>
(18)       <ProtocolAndModelClassifier>
(19)          http://schemas.xmlsoap.org/ws/2004/09/transfer
(20)       </ProtocolAndModelClassifier>
(21)       <ProtocolAndModelClassifier>
(22)          http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
(23)       </ProtocolAndModelClassifier>
(24)       <ProtocolAndModelClassifier>
(25)          http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer
(26)       </ProtocolAndModelClassifier>
(27)       <ProtocolAndModelClassifier>
(28)          http://example.com/product/xyzdevice/v1.0.2/component/cpu/PowerUp
(29)       </ProtocolAndModelClassifier>
(30)       <ProtocolAndModelClassifier>
(31)
     http://example.com/product/xyzdevice/v1.0.2/component/cpu/PowerDown
(32)       </ProtocolAndModelClassifier>
(33)       <Reference>
(34)         <MetaEPR AddressingVersions="
(35)               http://schemas.xmlsoap.org/ws/2005/08/addressing
(36)               http://schemas.xmlsoap.org/ws/2004/08/addressing">
(37)           <ParameterMap>
(38)             <Parameter Token="server" QNameType="innerValueOfGED"
(39)                QName="Host"/>
(40)           </ParameterMap>
```

```
(41)           <Address> http://{server}/mgmt/cpu </Address>
(42)         </MetaEPR>
(43)       </Reference>
(44)     </ResourceRef>
(45)     <ResourceRef>
(46)       <ResourceElement
(47)          Namespace="http://example.com/product/xyzdevice/v1.0.2/device.xsd"
(48)          LocalName="ComputerSystem"/>
(49)       <ProtocolAndModelClassifier>
(50)          http://schemas.xmlsoap.org/ws/2004/09/transfer
(51)       </ProtocolAndModelClassifier>
(52)       <ProtocolAndModelClassifier>
(53)          http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
(54)       </ProtocolAndModelClassifier>
(55)       <ProtocolAndModelClassifier>
(56)          http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer
(57)       </ProtocolAndModelClassifier>
(58)       <ProtocolAndModelClassifier>
(59)          http://example.com/product/xyzdevice/v1.0.2/component/cpu/PowerUp
(60)       </ProtocolAndModelClassifier>
(61)       <ProtocolAndModelClassifier>
(62)
    http://example.com/product/xyzdevice/v1.0.2/component/cpu/PowerDown
(63)       </ProtocolAndModelClassifier>
(64)       <Reference>
(65)         <MetaEPR AddressingVersions="
(66)               http://schemas.xmlsoap.org/ws/2005/08/addressing
(67)               http://schemas.xmlsoap.org/ws/2004/08/addressing">
(68)           <ParameterMap>
(69)             <Parameter Token="server" QNameType="innerValueOfGED"
(70)                QName="Host"/>
(71)           </ParameterMap>
(72)           <Address> https://{server}/mgmt/cpu </Address>
(73)         </MetaEPR>
(74)       </Reference>
(75)     </ResourceRef>
(76)   </Resource>
(77)   <EntryRef
(78)   Role="http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog/roles/child">
(79)     <EntryId>
(80)       http://example.com/product/xyzdevice/v1.0.2/catalog.xml#sensors
(81)     </EntryId>
(82)   </EntryRef>
(83)   <EntryRef
(84)   Role="http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog/roles/child">
(85)     <EntryId>
(86)       http://example.com/product/xyzdevice/v1.0.2/catalog.xml#eventLog
(87)     </EntryId>
(88)   </EntryRef>
```

```
(89) </Entry>
```

Notes:

- Lines (05)-(06) classify this entry as being about hardware and power management.
- This entry is a starting point for navigation based upon the Display Root classifier on line (08). It links to the sensor and event log entries on lines (80) and (86) respectively.
- The computer system resource can be accessed over either HTTP or HTTPS so there are two ResourceRef elements starting on lines (14) and (45) respectively.
- The XML representation of the resource uses the ComputerSystem GED as indicated on lines (15)-(17).
- The resource supports the Get operation from WS-Transfer and WS-ResourceTransfer to retrieve its current state. This is indicated by the classifiers on lines (18)-(26).
- The put operation is not supported so it is not advertised.
- The resource provides two methods for turning the machine on or off as indicated by the action URIs used as classifiers on lines (28) and (31).
- The resource can be accessed using either version of WS-Addressing as indicated on lines (35)-(36).

I.A.2 Sensor Entry

This entry represents the collection of sensors on the device that can monitor the CPU and other components. The number of sensors varies based upon how the device is wired to the rest of the computer so individual sensors cannot be listed in the catalog published on the web site. Instead, this entry contains a ResourceRef that allows for iterating all sensors as well as a ResourceRef that accesses a given sensor by its numeric id.

```
(90) <Entry
(91)       Id="http://example.com/product/xyzdevice/v1.0.2/catalog.xml#sensors">
(92)   <Descriptor>
(93)     <DisplayName xml:lang="en-US"> Sensors </DisplayName>
(94)   </Descriptor>
(95)   <Classifier> http://example.com/classifiers/hardware </Classifier>
(96)   <Classifier> http://example.com/classifiers/sensors </Classifier>
(97)   <Annotation xml:lang="en-US"> Sensor </Annotation>
(98)   <Annotation xml:lang="en-US"> Temperature </Annotation>
(99)   <Resource>
(100)      <ResourceRef>
(101)        <ResourceElement
(102)         Namespace="http://example.com/product/xyzdevice/v1.0.2/device.xsd"
(103)         LocalName="Sensor"/>
(104)        <ProtocolAndModelClassifier>
(105)            http://schemas.xmlsoap.org/ws/2004/09/enumerate/Enumerate
(106)        </ProtocolAndModelClassifier>
```

```
(107)         <Reference>
(108)           <MetaEPR AddressingVersions="
(109)                 http://schemas.xmlsoap.org/ws/2005/08/addressing
(110)                 http://schemas.xmlsoap.org/ws/2004/08/addressing">
(111)             <ParameterMap>
(112)               <Parameter Token="server" QNameType="innerValueOfGED"
(113)                  QName="Host"/>
(114)             </ParameterMap>
(115)             <Address> http://{server}/mgmt/sensor </Address>
(116)           </MetaEPR>
(117)         </Reference>
(118)       </ResourceRef>
(119)       <ResourceRef>
(120)         <ResourceElement
(121)
    Namespace="http://example.com/product/xyzdevice/v1.0.2/device.xsd"
(122)           LocalName="Sensor"/>
(123)         <ProtocolAndModelClassifier>
(124)           http://schemas.xmlsoap.org/ws/2004/09/enumerate/Enumerate
(125)         </ProtocolAndModelClassifier>
(126)         <Reference>
(127)           <MetaEPR AddressingVersions="
(128)                 http://schemas.xmlsoap.org/ws/2005/08/addressing
(129)                 http://schemas.xmlsoap.org/ws/2004/08/addressing">
(130)             <ParameterMap>
(131)               <Parameter Token="server" QNameType="innerValueOfGED"
(132)                  QName="Host"/>
(133)             </ParameterMap>
(134)             <Address> https://{server}/mgmt/sensor </Address>
(135)           </MetaEPR>
(136)         </Reference>
(137)       </ResourceRef>
(138)       <ResourceRef>
(139)         <ResourceElement
(140)          Namespace="http://example.com/product/xyzdevice/v1.0.2/device.xsd"
(141)          LocalName="Sensor"/>
(142)         <ProtocolAndModelClassifier>
(143)           http://schemas.xmlsoap.org/ws/2004/09/transfer
(144)         </ProtocolAndModelClassifier>
(145)         <ProtocolAndModelClassifier>
(146)           http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
(147)         </ProtocolAndModelClassifier>
(148)         <ProtocolAndModelClassifier>
(149)           http://schemas.xmlsoap.org/ws/2004/09/transfer/Put
(150)         </ProtocolAndModelClassifier>
(151)         <ProtocolAndModelClassifier>
(152)           http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer
(153)         </ProtocolAndModelClassifier>
(154)         <Reference>
```

```
(155)          <MetaEPR AddressingVersions="
(156)                http://schemas.xmlsoap.org/ws/2005/08/addressing
(157)                http://schemas.xmlsoap.org/ws/2004/08/addressing">
(158)            <ParameterMap>
(159)              <Parameter Token="server" QNameType="innerValueOfGED"
(160)                  QName="Host"/>
(161)              <Parameter Token="ID" QNameType="simpleType"
(162)                  QName="xs:integer">
(163)                <Description xml:lang="en-US">
(164)                  The id of the sensor
(165)                </Description>
(166)                <Example> 7 </Example>
(167)              </Parameter>
(168)            </ParameterMap>
(169)            <Address> http://{server}/mgmt/sensor </Address>
(170)            <ReferenceParameters>
(171)              <vendor:SensorId> {ID} </vendor:SensorId>
(172)           </ReferenceParameters>
(173)          </MetaEPR>
(174)        </Reference>
(175)      </ResourceRef>
(176)      <ResourceRef>
(177)        <ResourceElement
(178)         Namespace="http://example.com/product/xyzdevice/v1.0.2/device.xsd"
(179)         LocalName="Sensor"/>
(180)        <ProtocolAndModelClassifier>
(181)          http://schemas.xmlsoap.org/ws/2004/09/transfer
(182)        </ProtocolAndModelClassifier>
(183)        <ProtocolAndModelClassifier>
(184)          http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
(185)        </ProtocolAndModelClassifier>
(186)        <ProtocolAndModelClassifier>
(187)          http://schemas.xmlsoap.org/ws/2004/09/transfer/Put
(188)        </ProtocolAndModelClassifier>
(189)        <ProtocolAndModelClassifier>
(190)          http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer
(191)        </ProtocolAndModelClassifier>
(192)        <Reference>
(193)          <MetaEPR AddressingVersions="
(194)                http://schemas.xmlsoap.org/ws/2005/08/addressing
(195)                http://schemas.xmlsoap.org/ws/2004/08/addressing">
(196)            <ParameterMap>
(197)              <Parameter Token="server" QNameType="innerValueOfGED"
(198)                  QName="Host"/>
(199)              <Parameter Token="ID" QNameType="simpleType"
(200)                  QName="xs:integer">
(201)                <Description xml:lang="en-US">
(202)                  The id of the sensor
(203)                </Description>
```

```
(204)                <Example> 7 </Example>
(205)              </Parameter>
(206)            </ParameterMap>
(207)            <Address> https://{server}/mgmt/sensor </Address>
(208)          <ReferenceParameters>
(209)              <vendor:SensorId> {ID} </vendor:SensorId>
(210)            </ReferenceParameters>
(211)          </MetaEPR>
(212)        </Reference>
(213)      </ResourceRef>
(214)    </Resource>
(215)  </Entry>
```

Notes:

- The ResourceRefs on lines (100) and (119) are iterators (using HTTP and HTTPS).
    - All sensors can be retrieved using WS-Enumeration as advertised on line (105).
    - The MetaEPRs for HTTP and HTTPS are both parameterized by the host name of the device as in the ComputerSystem Entry.
- The ResourceRefs on lines (119) and (176) are accessors (using HTTP and HTTPS)
    - A given sensor can be retrieved using WS-Transfer/WS-ResourceTransfer as advertised on lines (142)-(153). The same dialects are supported as for the ComputerSystem.
    - The sensor id is included in the ParameterMap for the MetaEPR on line (161) and the value is included as a reference parameter on line (171).
    - A sensor can be updated via the Put operation from WS-Transfer as advertised on line (149).

I.A.3 EventLog Entry

This entry represents the log of sensor events stored on the device. Sensors write records in the event log when thresholds are exceeded.

```
(216)  <Entry
(217)     Id="http://example.com/product/xyzdevice/v1.0.2/catalog.xml#eventLog">
(218)    <Descriptor>
(219)      <DisplayName xml:lang="en-US"> Event Log </DisplayName>
(220)    </Descriptor>
(221)    <Classifier> http://example.com/classifiers/hardware </Classifier>
(222)    <Classifier> http://example.com/classifiers/events </Classifier>
(223)    <Resource>
(224)      <ResourceRef>
(225)        <ResourceElement
(226)          Namespace="http://example.com/product/xyzdevice/v1.0.2/device.xsd"
(227)          LocalName="EventLogEntry"/>
```

```
(228)        <ProtocolAndModelClassifier>
(229)          http://schemas.xmlsoap.org/ws/2004/09/enumerate/Enumerate
(230)        </ProtocolAndModelClassifier>
(231)        <ProtocolAndModelClassifier>
(232)          http://schemas.xmlsoap.org/ws/2004/09/eventing/Subscribe
(233)        </ProtocolAndModelClassifier>
(234)        <ProtocolAndModelClassifier>
(235)
    http://example.com/product/xyzdevice/v1.0.2/component/eventLog/Clear
(236)        </ProtocolAndModelClassifier>
(237)        <Reference>
(238)          <MetaEPR AddressingVersions="
(239)                http://schemas.xmlsoap.org/ws/2005/08/addressing
(240)                http://schemas.xmlsoap.org/ws/2004/08/addressing">
(241)            <ParameterMap>
(242)              <Parameter Token="server" QNameType="innerValueOfGED"
(243)                 QName="Host"/>
(244)            </ParameterMap>
(245)            <Address> http://{server}/mgmt/eventlog </Address>
(246)          </MetaEPR>
(247)        </Reference>
(248)      </ResourceRef>
(249)      <ResourceRef>
(250)        <ResourceElement
(251)         Namespace="http://example.com/product/xyzdevice/v1.0.2/device.xsd"
(252)         LocalName="EventLogEntry"/>
(253)        <ProtocolAndModelClassifier>
(254)          http://schemas.xmlsoap.org/ws/2004/09/enumerate/Enumerate
(255)        </ProtocolAndModelClassifier>
(256)        <ProtocolAndModelClassifier>
(257)          http://schemas.xmlsoap.org/ws/2004/09/eventing/Subscribe
(258)        </ProtocolAndModelClassifier>
(259)        <ProtocolAndModelClassifier>
(260)
    http://example.com/product/xyzdevice/v1.0.2/component/eventLog/Clear
(261)        </ProtocolAndModelClassifier>
(262)        <Reference>
(263)          <MetaEPR AddressingVersions="
(264)                http://schemas.xmlsoap.org/ws/2005/08/addressing
(265)                http://schemas.xmlsoap.org/ws/2004/08/addressing">
(266)            <ParameterMap>
(267)              <Parameter Token="server" QNameType="innerValueOfGED"
(268)                 QName="Host"/>
(269)            </ParameterMap>
(270)            <Address> https://{server}/mgmt/eventlog </Address>
(271)          </MetaEPR>
(272)        </Reference>
(273)      </ResourceRef>
(274)    </Resource>
```

```
(275) </Entry>
```

Notes:

- The messages in the EventLog can be retrieved via WS-Enumeration as indicated by line (229).
- New messages written to the EventLog can be sent when they occur by using WS-Eventing as indicated on line (232).
- Because the log entries can be enumerated, the ResourceElement on lines (225)-(227) indicates that the items returned use the EventLogEntry GED.
- All messages in the EventLog can be erased by the custom method advertised on line (235).
- The EventLog resource can be accessed over either HTTP or HTTPS so there are two ResourceRef elements starting on lines (224) and (249) respectively.

I.B Software Service Catalog

The following catalog represents a Web service that can be managed using Web service management protocols. This catalog is shown as an XML document but, as mentioned in section 4.1.2, a catalog is not always retrieved as a whole. The same information can also be retrieved through query and/or enumeration interfaces over the sequence of Entry elements in the catalog. It is likely that a catalog with a large number of Entry elements would be accessed in this manner.

The management endpoint reports the manageable state of a Web service providing stock quotes. The management resource supports WS-ResourceTransfer as shown by the protocol classifiers on lines (17)-(25) The WSDL representing the interface to the management resource is included as in-lined metadata on lines (34)-(83). Line (90) includes one property from the resource representation to aid in discovering this particular Web service among others in the same catalog.

```
(01) <wsrc:Entry Id="http://example.com/ManagementEndpoint2"
(02)     xmlns:wsrc="http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog"
(03)     xmlns:mex="http://schemas.xmlsoap.org/ws/2004/09/mex"
(04)     xmlns:wsa="http://www.w3.org/2005/08/addressing">
(05)   <wsrc:Descriptor>
(06)     <wsrc:DisplayName xml:lang="en-US">
(07)       Management endpoint for stock quote service.
(08)     </wsrc:DisplayName>
(09)   </wsrc:Descriptor>
(10)   <wsrc:Annotation xml:lang="en-US">software service</wsrc:Annotation>
(11)
(12)   <wsrc:Resource>
(13)     <wsrc:ResourceRef>
(14)       <wsrc:ResourceElement
(15)         LocalName="StockQuoteManagementServiceProperties"
(16)         Namespace="http://example.org/StockQuoteManagementMetrics" />
(17)       <wsrc:ProtocolAndModelClassifier>
```

```
(18)          http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer
(19)        </wsrc:ProtocolAndModelClassifier>
(20)        <wsrc:ProtocolAndModelClassifier>
(21)          http://schemas.xmlsoap.org/ws/2004/09/transfer
(22)        </wsrc:ProtocolAndModelClassifier>
(23)        <wsrc:ProtocolAndModelClassifier>
(24)          http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
(25)        </wsrc:ProtocolAndModelClassifier>
(26)        <wsrc:Reference>
(27)          <wsa:EndpointReference>
(28)            <wsa:Address>
(29)              http://example.org/services/StockQuoteManagementServiceEndpoint
(30)            </wsa:Address>
(31)          </wsa:EndpointReference>
(32)        </wsrc:Reference>
(33)
(34)        <mex:Metadata>
(35)          <mex:MetadataSection Dialect="http://schemas.xmlsoap.org/wsdl/">
(36)            <wsdl:definitions
(37)              targetNamespace="http://example.org/services/stockQuote"
(38)              xmlns:xs="http://www.w3.org/2001/XMLSchema"
(39)              xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
(40)              xmlns:soapwsdl="http://schemas.xmlsoap.org/wsdl/soap/"
(41)              xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
(42)              xmlns:tns="http://example.org/services/stockQuote"
(43)
    xmlns:wsrt="http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer">
(44)
(45)            <wsdl:import
(46)
    namespace="http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer"
(47)             location="wsrt.wsdl" />
(48)
(49)            <wsdl:portType
(50)              name="StockQuoteManagementServicePortType">
(51)              <wsdl:operation name="Get">
(52)                <wsdl:input name="Get"
(53)                  message="wsrt:GetRequestMessage"
(54)                  wsa:Action=
(55)                    "http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
(56)                <wsdl:output
(57)                  name="GetResourcePropertyResponse"
(58)                  message="wsrt:GetResponseMessage"
(59)                  wsa:Action=
(60)
    "http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse"/>
(61)                <!--   Faults removed to compact example  -->
(62)              </wsdl:operation>
(63)            </wsdl:portType>
(64)
```

```
(65)           <wsdl:binding
(66)             name="StockQuoteManagementServiceBinding"
(67)             type="StockQuoteManagementServicePortType">
(68)
(69)             <!--   Bindings removed to compact example  -->
(70)
(71)           </wsdl:binding>
(72)
(73)           <wsdl:service name="StockQuoteManagementService">
(74)             <wsdl:port
(75)               name="StockQuoteManagementServicePort"
(76)               binding="StockQuoteManagementServiceBinding">
(77)               <soapwsdl:address
(78) location="http://example.org/services/StockQuoteManagementServiceEndpoint"
     />
(79)             </wsdl:port>
(80)           </wsdl:service>
(81)           </wsdl:definitions>
(82)
(83)         </mex:MetadataSection>
(84)       </mex:Metadata>
(85)     </wsrc:ResourceRef>
(86)
(87)     <wsrc:ResourceDiscoveryProperties
(88)          xmlns:mgmt="http://example.org/StockQuoteManagementMetrics">
(89)       <mgmt:ResourceId>
(90)         urn:uuid:f51b72f5-1163-4388-8634-b7d08cb7341a
(91)       </mgmt:ResourceId>
(92)     </wsrc:ResourceDiscoveryProperties>
(93)   </wsrc:Resource>
(94) </wsrc:Entry>
```

Appendix II – XML Schema

A normative copy of the XML Schema [XML Schema Part 1, Part 2] description for this specification can be retrieved from the following address:

```
http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog
```

A non-normative copy of the XML Schema description is listed below for convenience.

```
<?xml version="1.0" ?>
<!--
Copyright Notice
(c) 2007 Hewlett-Packard Development Company (HP), Intel Corporation,
International Business Machines Corporation (IBM), and Microsoft
Corporation. All rights reserved.

Permission to copy and display the "Web Services Resource Catalog"
Specification, in any medium without fee or royalty is hereby granted,
provided that you include the following on ALL copies of the "Web
```

```
Services Resource Catalog" Specification, or portions thereof, that
you make:
 1. A link or URL to the "Web Services Resource Catalog"
    Specification at this location:
       http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog
 2. The copyright notice as shown in the "Web Services Resource
    Catalog" Specification.

Hewlett-Packard Development Company (HP), Intel Corporation,
International Business Machines Corporation (IBM), and Microsoft
Corporation (collectively, the "Authors") each agree to grant you a
royalty-free license, under reasonable, non-discriminatory terms and
conditions to their respective patents that they deem necessary to
implement the "Web Services Resource Catalog" Specification.

THE "WEB SERVICES RESOURCE CATALOG" SPECIFICATION IS PROVIDED "AS IS,"
AND THE AUTHORS 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 "WEB SERVICES RESOURCE CATALOG" 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 AUTHORS 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 "WEB SERVICES RESOURCE CATALOG"
SPECIFICATION.

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

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

<xs:schema xmlns:tns="http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       targetNamespace="http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog"
       elementFormDefault="qualified">
    <xs:import namespace="http://www.w3.org/XML/1998/namespace"
               schemaLocation="http://www.w3.org/2001/xml.xsd"/>
    <!-- Constructs from the WS-Addressing Core adapted to MetaEPR -->
    <xs:complexType name="MetaEPRType" mixed="false">
        <xs:sequence>
            <xs:element ref="tns:ParameterMap" minOccurs="0"/>
```

```
            <xs:element name="Address" type="xs:string"/>
            <xs:element name="ReferenceParameters"
type="tns:MetaEndpointElementType" minOccurs="0"/>
            <xs:element name="Metadata" type="tns:MetaEndpointElementType"
minOccurs="0"/>
            <xs:element name="Any" type="tns:MetaEndpointExtensibilityType"
minOccurs="0"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="AddressingVersions" type="tns:URIListType"
use="required"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:simpleType name="URIListType">
        <xs:list itemType="xs:anyURI"/>
    </xs:simpleType>
    <xs:complexType name="MetaEndpointElementType" mixed="true">
        <xs:sequence>
            <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="MetaEndpointExtensibilityType" mixed="true">
        <xs:sequence>
            <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <!-- End -->
    <xs:element name="ParameterMap" type="tns:ParameterMapType"/>
    <xs:complexType name="ParameterMapType" mixed="false">
        <xs:sequence>
            <xs:element name="Parameter" type="tns:ParameterType"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="ParameterType" mixed="false">
        <xs:sequence>
            <xs:element name="Description" type="tns:LocalizableStringType"
minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Example" type="tns:ExampleType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="Token" type="xs:NCName" use="required"/>
        <xs:attribute name="QNameType" type="tns:QNameTypeType"
use="required"/>
        <xs:attribute name="QName" type="xs:QName" use="required"/>
```

```
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:simpleType name="QNameTypeType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="simpleType"/>
            <xs:enumeration value="innerValueOfGED"/>
            <xs:enumeration value="outerValueOfGED"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="ExampleType" mixed="true">
        <xs:sequence>
            <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="MetaURIType" mixed="false">
        <xs:sequence>
            <xs:element ref="tns:ParameterMap"/>
            <xs:element name="TemplateURI" type="xs:string"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="DescriptorType">
        <xs:sequence>
            <xs:element name="DisplayName" type="tns:LocalizableStringType"
minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Publisher" type="xs:string" minOccurs="0"/>
            <xs:element name="PublisherURL" type="xs:anyURI" minOccurs="0"/>
            <xs:element name="ResourceURL" type="xs:anyURI" minOccurs="0"/>
            <xs:element name="Version" type="xs:string" minOccurs="0"/>
            <xs:element name="Created" type="xs:dateTime" minOccurs="0"/>
            <xs:element name="Updated" type="xs:dateTime" minOccurs="0"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="ClassifierType">
        <xs:simpleContent>
            <xs:extension base="xs:anyURI">
                <xs:anyAttribute namespace="##other" processContents="lax"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="ResourceType">
        <xs:sequence>
            <xs:element name="ResourceRef" type="tns:ResourceRefType"
maxOccurs="unbounded"/>
            <xs:element name="ResourceDiscoveryProperties" minOccurs="0">
```

```
        <xs:complexType>
          <xs:sequence>
            <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:complexType>
            </xs:element>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="ResourceElementType">
        <xs:attribute name="Namespace" type="xs:anyURI" use="required"/>
        <xs:attribute name="LocalName" type="xs:NCName" use="required"/>
    </xs:complexType>
    <xs:complexType name="ReferenceType">
            <xs:choice>
                <xs:element name="URI" type="xs:anyURI"/>
                <!-- a WS-Addressing EPR -->
                <xs:any namespace="##other" processContents="lax"/>
            </xs:choice>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="ParameterizableReferenceType">
            <xs:choice>
                <xs:element name="URI" type="xs:anyURI"/>
                <xs:element name="MetaURI" type="tns:MetaURIType"
maxOccurs="unbounded"/>
                <xs:element name="MetaEPR" type="tns:MetaEPRType"
maxOccurs="unbounded"/>
                <!-- a WS-Addressing EPR -->
                <xs:any namespace="##other" processContents="lax"/>
            </xs:choice>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="ResourceRefType">
        <xs:sequence>
            <xs:element name="ResourceElement" type="tns:ResourceElementType"
minOccurs="0"/>
          <xs:element name="ProtocolAndModelClassifier"
type="tns:ClassifierType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Reference"
type="tns:ParameterizableReferenceType"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="EntryReferenceType">
```

```
        <xs:sequence>
            <xs:element name="EntryId" type="xs:anyURI"/>
            <xs:element name="RemoteRef" type="tns:RemoteRefType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="Role" type="xs:anyURI" use="required"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="RemoteRefType">
        <xs:sequence>
            <xs:element name="ProtocolClassifier" type="tns:ClassifierType"
minOccurs="0"/>
            <xs:element name="Reference" type="tns:ReferenceType"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="RefType" type="tns:RefTypeType" use="required"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:simpleType name="RefTypeType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Catalog"/>
            <xs:enumeration value="Entry"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:element name="Entry" type="tns:EntryType"/>
    <xs:complexType name="EntryType">
        <xs:sequence>
            <xs:element name="Descriptor" type="tns:DescriptorType"
minOccurs="0"/>
            <xs:element name="Classifier" type="tns:ClassifierType"
minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Annotation" type="tns:LocalizableStringType"
minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Resource" type="tns:ResourceType" minOccurs="0"/>
            <xs:element name="EntryRef" type="tns:EntryReferenceType"
minOccurs="0" maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="Id" type="xs:anyURI" use="required"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="CatalogType">
        <xs:sequence>
            <xs:element ref="tns:Entry" minOccurs="0" maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
```

```
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:element name="Catalog" type="tns:CatalogType"/>
    <xs:complexType name="LocalizableStringType">
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute ref="xml:lang" use="optional"/>
          </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!-- GEDs for use in ParameterMap -->
    <xs:element name="Host" type="xs:string"/>
    <xs:element name="Port" type="xs:positiveInteger"/>
</xs:schema>
```