WEB SERVICES STANDARD-BASED SYSTEM RESOURCE MANAGEMENT MIDDLEWARE MODEL, SCHEME AND TEST

Zhihui Lu¹, Member, IEEE, Jie Wu², and Patrick Hung³, Member, IEEE
¹School of Computer Science, Fudan University, Shanghai, China,
²Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, China
³University of Ontario Institute of Technology, Canada
lzh,jwu@fudan.edu.cn, patrick.hung@uoit.ca

Abstract
Traditionally, system resource management software is tightly tied with managed IT resources through their specific manageability interfaces. Applying Web Services technology to the field of system resource management is a reasonable way to loosen this tie. In this paper, we discuss the implementation of SOA principals to the system resource management field, and then use WSDM standard and WS-Management standards to design and realize the resource management middleware model. We illustrate how Web Services management standards are mapped and integrated with existing management interfaces, such as JMX, WMI, and so on. As part of our experimental work, we discuss MUSE-JMX and Wiseman-WMI based system management implementation schemes, including Manager Layer, Gateway Layer, and Resource Agent Layer, and then we analyze the experiments results. After that, from our research experiences and related surveys, we analyze the gap between WS-Management and real management fields and design some feasible solution for these gaps. Finally, We compare WSDM and WS-Mangement standard stack, and then we discuss the prospective research direction and challenges in this field.

Keywords: System Resource Management, Web Services, SOA, WSDM, WS-Management, Wiseman, JMX, WMI

1. INTRODUCTION

System resource management applications (SRMA, for short) have a lot of capabilities that manage IT resources, such as hardware management, performance management, desktop management, etc. To achieve these management targets, the traditional way is to use specific and various manageability interfaces to communicate with the managed IT resources directly or through agents. All kinds of technologies of manageability interfaces vary from resources to resources, from agents to agents and also from resources to agents. Traditional system resource management is demonstrated in Fig. 1.

The management system is the place where SRMAs are deployed. It has all the management policies and logic to manage the whole resource (including hardware, software, and all other IT resources). Based on management policies and logic, it helps system administrators to manage the whole system conveniently, efficiently and effectively, and carry out management orders by taking advantage of manageability interfaces of managed IT resources.

Lines labeled with numbers in Fig. 1 represent communication between SRMA and the manageability interfaces of managed IT resources. Because many different IT resources have different manageability interfaces, it makes a SRMA tightly coupled with specific manageability interfaces of the managed IT resources, and vulnerable to the changes of managed IT resources’ manageability interfaces. Therefore, it is difficult to build a robust and effective application to manage so many heterogeneous IT resources. The reason is that there are too many specific manageability interfaces for a single system resource management application to cover them all. When the SRMA system is deployed, it is hard to install a new manageable IT resource without requiring recompilation of the SRMA or bringing in new security threats.

Fig. 1. Traditional System Resource Management Model

Specifically, when we install new managed IT resources behind firewalls into our SRMAs, it may introduce new security threats to our existing system.
2. RELATED WORKS

In recent years, Service Oriented Architecture (SOA) has become highly recognized as a system resource management technology. As one representative of SOA, Web Services technology and standards are more and more used in the system resource management field.

Web Service Distributed Management standard specification (WSDM [1, 2]), as an initiative of OASIS organization, helps build the SOA-supported manageability interfaces of IT resources. By leveraging web service technology, WSDM enables easier and more efficient resource management. It provides a flexible and common framework for manageability interfaces, and can be used to achieve universal management and interoperability across many various types of distributed IT resources.

Distributed Management Task Force (DMTF) published another management standard specification “Web Service for Management” (WS-Management) [3], which describes a general Web Services-based protocol for managing systems such as PCs, servers, devices, web services, other applications and other manageable entities.

In paper [5], we proposed SOAMS—a novel SOA-based system and network management scheme and a WSDM-based management middleware model for SOAMS. As part of our experiments work, we discuss a MUSE-JMX based management scheme implementation—SOAMS-Platform, including Manager Layer, WSDM-Gateway Layer, and Agent Layer, and then analyze the prospective research direction. In paper [10], we also focus on the WSDM-based management middleware model, including the programming model for the WSDM-based management middleware layer, and the protocol hierarchy in the middleware layer. In paper [13], we emphasized heterogeneous resource and service status monitoring and management issues; we further propose a novel WSRF-WSDM and multi-agent based distributed system resource management scheme—WSDM-RMS. Xuan T. N. et al. [18] proposes an extension of existing Web services-agent integration toolkit WS2JADE for Web services management, especially using the WSDM interface.

In paper [6], based on WS-Management, we proposed a distributed system and network resource management middleware model. In this model, every managed IT resource provides manageability interfaces via the WS-Management specification. Furthermore, we utilize the WS-Management Java implementation prototype-wiseman and the WMI management interface to carry out the scheme implementation and test case work of the novel model, and then analyze the experiment results. In paper [11], we described a novel WS-Management-based system resource management middleware model and scheme. In paper [37], we present three new test evaluation schemes for WS-Management protocol and its implementation project-Wiseman, including JMeter & JProbe based semi-automatic test scheme, Grinder & JProbe based automatic test scheme, and Wiseman and WMI based test scheme. This paper extends our paper [5][6][10][11][37]’s work.

Ricardo L.V et al. [14] showed that Web services technologies have more to offer to the network management discipline than just bridging established network management protocols and Web services protocols. Particularly they explore the possibility of using Web services composition applied to network management. Herwig Manaaert et al. [15] investigated the use of web services as an open communication standard for network management systems. After a concise description of web services and their possible advantages for network management systems, they present a case study. A. Pras et al. [16] and T. Fioreze et al. [17] compared Web Services with SNMP in a System and Network Management environment.

Although the web service standards provide a loosely coupled management interface, there already are a large number of management interfaces, so we need to study how to map existing heterogeneous management interfaces to the web service management interfaces. In this paper, we discuss the implementation of the SOA principal in the system resource management field. To summarize, the key contributions and key findings of this paper are the following:

We use the WSDM standard and the WS-Management standard to design and realize the resource management middleware model and implementation scheme.

Specifically, we illustrate how Web Services management standards are mapped and integrated with other management interfaces, such as JMX, WMI, and so on. Based on the WSDM Middleware Model, we have developed and implemented a MUSE and JMX based resource management scheme. Based on the WS-Management Middleware Model, we have developed and implemented a Wiseman and WMI based resource management scheme.

Through these models, we transform the specific manageability interfaces to web service standard based manageability interfaces, and achieve the object of universal management and interoperability of distributed IT resources. We have tested and verified these schemes’ function and performance.

The rest of this paper is organized as follows: In Sec. 3, we introduce the basic architecture of Web Services-based system resource management. We propose WSDM -based System Resource Management Middleware Model and Implementation Scheme in Sec.4. WS-Management-based System Resource Management Middleware Model and Implementation Scheme is shown and discussed in Sec.5. We compare WSDM and WS-Management standard stack in Sec.6. Our conclusion and prospecting remarks are given in Sec.7.

3. WEB SERVICES-BASED SYSTEM RESOURCE MANAGEMENT ARCHITECTURE

In the system resource management field, we can use many kinds of technologies in managed IT resources to sup-
port SOA-based SRMAs. Web service is a reasonable and representative choice, as the web service related management standard requires standardized and flexible manageability interfaces. As Fig. 2 demonstrates, we propose a Web Services-based system resource management architecture.

In the Web Services-based SRMA, every managed IT resource provides its manageability interfaces via Services. These interfaces are standardized and independent of the special platform where the services are deployed. Based on the common framework of manageability interfaces, managing various IT resources becomes easier through a universal, standard way. Also SRMAs become easier to evolve into more robust and powerful ones without the limit imposed by the close ties between SRMAs and specific manageability interfaces of managed IT resources.

Just as Fig. 2 illustrates, in this architecture, some managed IT resource can directly expose their manageability interfaces via web services, and some resources still use the existing private interfaces, but they also can expose new managable web services interfaces through the management gateway. The management gateway is responsible for mapping multiple resources to the manageable web services. The management system can access manageable web services directly, or through the gateway, and carry out all kinds of management operations, such as monitoring performance or subscribing to events. Because all the requests of the management system are based on standard web service protocols, the management system is loosely coupled with the managed IT resources. If the newly added managed resources have standard web service interfaces, the management system can easily access and manage them.

In the following section, we propose a WSDM standard and WS-Management standard-based system resource management middleware model. Specifically, we focus on the main mechanisms of Web Services management standards mapping and integration mechanism with other existing heterogeneous management interfaces, such as JMX, WMI, SNMP et al.

4. WSDM-BASED SYSTEM RESOURCE MANAGEMENT MIDDLEWARE MODEL AND SCHEME

In the system resource management field, we can use many kinds of SOA technologies in managed IT resources to support SOA-based SRMAs. WSDM provides standardized manageability interfaces to support SOA-based SRMAs via web services.

4.1. WSDM Introduction

From Fig. 3, we can see the WSDM protocol hierarchy from bottom to top.

SOAP, WSDL: The fundamental protocols for using web services. SOAP is a transport layer to send messages between the service consumer and the service provider. WSDL is used to describe the service. A service consumer can get the WSDL for the service that has the description, and invoke the service using SOAP.

WS-Addressing (WSA) [19]: provides transport-neutral mechanisms to address Web services and messages.

WS-Resource Framework (WSRF) [20-22]: define an open framework for modeling and accessing stateful resources using Web services.

WS-Metadata Exchange (WS-MEX) [23]: defines messages to retrieve metadata associated with an endpoint.

WS-Notification (WSN) [24]: advance a pattern-based approach to allow Web services to disseminate information to each other, especially subscription and eventing.

Acting as a resource virtualization engine, WSRF provides standardized mechanisms for associating XML-based resource properties with network entities and for accessing those properties via either pull (query) or push (subscription). WSRF provides two aggregator services that collect recent
state information from registered information sources. These different mechanisms provide a powerful framework for monitoring diverse collections of distributed components and for obtaining information about components for purposes of discovery. Based on WSRF, WSN and WS-MEX, the OASIS Web Services Distributed Management TC has defined two sets of specifications: Web Services Distributed Management: Management Using Web Services (MUWS) [1], and Management Of Web Services (MOWS) [2]. The MUWS specifications define how to use Web services to expose manageable resources, and the MOWS specifications define how to expose manageable Web service implementations. Application of MOWS to MUWS gives an interesting combination of the manageable interface [2]. OASIS WSDM V1.1 has been approved as an OASIS standard on October 4, 2006.

WSDM further exposes the Web Service Endpoint of the system resource, as the manageability interface. The manageability interfaces support common operations: (a) Control: start, stop, etc; (b) Monitoring: status and performance; (c) Subscription and Notification: event.

The basic concepts of WSDM can be illustrated by the following Fig. 4.

![Fig. 4. WSDM Concept](image)

A Web service endpoint provides access to a manageable resource [1]. An example of a manageable resource is a printer that indicates when its toner is low, or a magnetic storage disk that reports its internal temperature. A manageability consumer discovers the Web service endpoint and exchanges messages with the endpoint in order to request information, subscribe to events, or, control the manageable resource associated with the endpoint. An example of a manageable consumer is a management system, a business automation process, or simply any Web service application.

The WSDM Specification [1, 2] defines the framework for managed IT resources to support Web Services-based SRMAs. It defines a way to build a manageability model via web services.

Based on WSDM, we designed SOAMS [5]: A Novel SOA-based System and Network Management Model and Scheme, and WSDSNM3 [10]: a Web Services-based Distributed System and Network Management Middleware Model.

We also illustrate the standard way of mapping system resources’ existing specific manageability interfaces to the manageability interfaces as web services.

### 4.2 Three Layer Architecture of WSDM-based System Resource Management Middleware Model

![Fig. 5. Three layer Architecture of WSDM-based System Resource Management Middleware Model](image)

Based on the distributed system management middleware model, the whole system can be divided logically into three layers: the Manager Layer, the Gateway Middleware Layer and the Managed Resource Layer. The three layer architecture is demonstrated in the Fig. 5.

Managed Resource Layer contains all the managed resources. These resources can be hardware or software and even services. To promote a consistent way to manage such abundant resource types, all resource management models should be standardized, like CIM [25] Core model, JMX [26], WMI [27], SNMP [28].

The managed system resource objects correspond to the WSDM Resource Instances that run in the Gateway Middleware Layer. These managed objects contain information on how to communicate with the corresponding WSDM Resource Instance. Manageability consumers (Manager Layer Components) use these to locate and manipulate the managed resources.

Gateway Middleware Layer provides standardized manageability interfaces for the Management Layer to manage disparate and diverse resources in the managed system layer. In our architecture, we utilize WSDM standard technology to expose unified interfaces for the management of underlying managed systems. Applying web service makes a good interoperability while managing different types of managed resources, since it integrates different manageability interfaces into a consistent manageability interface. Additionally, web services are inherently firewall-friendly, and this advantage relieves the heavy burden of configuration when the administrator needs to manage some resource behind the firewall,
which is a normal case in the realistic world. The Gateway Middleware Layer is based on WSDM specification. This gives a high guarantee to the interoperability of management among diverse and disparate IT resources.

Manager Layer analyzes the information that is gathered from the underlying managed system through the manageability interfaces provided by the Gateway Middleware Layer, then use a human-friendly way to show the running status of the managed system. The core of this layer is an intelligent and powerful information analysis mechanism and a human-friendly user operating interface.

The Manager Layer is a manageability consumer. It manipulates the manageable resource by invoking the WSDM management interface exposed by the Gateway Middleware Layer.

The main components of the Manager Layer carry out the following system management functions:

1) Configure System: Configure the parameters of a target application system, including setting the properties of the resources in the resource pool, and setting the threshold values of some scenarios, such as an overload condition.
2) View Metrics: View performance metrics of the manageable resource.
3) Subscribe and Receive Notification: When a management event is emitted, the manageability consumer who has subscribed to this event uses this operation to retrieve the event message.
4) Execute Management Operation: Execute custom-operations such as adding a new server when the service is overloaded.

4.3. MUSE and JMX based System Resource Management Scheme Implementation

![Diagram](image)

**Fig. 6. MUSE and JMX based System Resource Management Scheme**

Based on the WSDM Management Middleware Model, we have developed and implemented a prototype system: a MUSE and JMX based Resource Management Scheme, and we call it the SOAMS-Platform [5]. In the SOAMS-Platform, we have implemented resource monitoring and service scheduling using a JMX Agent technology and MUSE Project [30]. The system architecture is shown in Fig. 6.

Just as Fig. 6 describes, in this system, there are three parts: the Manager Layer, the MUSE-Gateway Layer, and the JMX-Agent Layer-the local managed resource layer. The MUSE-Gateway corresponds to the Gateway Middleware Layer, which is responsible for service status information collecting and event publishing, while the Manager Layer is in charge of retrieving status data from every resource service point, as well as managing and scheduling resources.

The SOAMS-Platform Manager Layer is developed using CIMOM-Common Information Model Object Manager [25]. It is the implementation of the core part of WBEM (DMTF - Web-Based Enterprise Management) protocols [29]. The DMTF Common Information Model (CIM) is an approach to the management of systems and networks that applies the basic structuring and conceptualization techniques of the object-oriented paradigm [25]. This approach uses a uniform modeling formalism that, together with the basic repertoire of object-oriented constructs-supports the cooperative development of an object-oriented schema across multiple organizations.

**Fig. 7. Class Diagram of Web Service Metrics**

The MUSE-Gateway Layer is developed based on the open source software 'MUSE 2.0', as its implementation of WSDM standards. MUSE [30] -is a Java-based implementation of the WS-Resource Framework (WSRF) 1.2, WS-Base Notification (WSN)1.3, and WS-Distributed Management (WSDM) 1.1 specifications. It is a framework upon which users can build web service interfaces for manageable resources without having to implement all of the “plumbing” described by the aforementioned standards. Applications built with Muse can be deployed in both Apache Axis2 and...
The project also includes a set of command line tools that can generate the proper artifacts for a specific deployment scenario.

The MUSE-Gateway Layer provides several manageable WSDM resource interfaces, including a Load Balancer, a Web Server, a Database Server and a Web Service. The Web Service itself acts as one kind of resource, mainly providing the MOWS interface. The class diagram of web service metrics is illustrated in Figure 7. The main attributes are:

1) The NumberOfRequests is the metric that is used to show the number of end-user requests to observe the load of the service.
2) The NumberOfSuccessfulRequests is the metric that is used to show the number of successful end-user requests to observe the QoS.
3) The NumberOfFailedRequests is the metric that is used to show the number of failed end-user requests to observe the QoS.
4) The MaxResponseTime is the metric that is used to observe the max response time of the service.
5) The AverageResponseTime is the metric that is used to show the average response time of the service. This metric is very important and valuable in measuring the load of the service.

The ManagerWebService Class has some operations that can be executed to react to certain management situations.

1) getServiceMetrics () : The operator can execute the getServiceMetrics method to get the metrics of the object service.
2) subscribeTopic () : The Operator can execute the subscribeTopic method to subscribe to some topics of the service.
3) receiveNotification () : The Operator can execute the receiveNotification method to receive the notification of the subscribed topics.
4) addANewServer () : The operator can execute the addANewServer method to add a new server to scale out the web tier of the system if the overload scenario happens. The operator should provide the parameter of the method such as the IP and the port number of the server.
5) stopNewLogin () : The Operator can execute the stopNewLogin to reject new login requests from end-users, if the overload scenario happens.

The Database Server acts as one kind of resource, mainly providing the MUWS interface. The class diagram of database server metrics is described in Figure 8.

The Agent Layer is a managed system resources layer, which can use its own management interface (such as JMX MBean, CIM interface, SNMP interface, etc.) to manipulate physical resources. We have designed and developed the following agents using JMX technology:

1) LB Server Agent
This agent should be able to retrieve the management metrics in the Load Balancer server, such as CPU utilization, memory utilization, network utilization, etc. It should also provide the management interface of the Load Balancer (e.g. add new server/remove server/stop new login).

2) Web Server Agent
This agent should be able to retrieve the management metrics in the Web Server, such as CPU utilization, memory utilization, network utilization, number of connections, etc. It should also provide an appropriate management interface for Web Server resources.

3) DB Server Agent
This agent should be able to retrieve the management metrics in Database Server machine such as CPU utilization, memory utilization, network utilization, number of connections, etc. It should also provide an appropriate management interface for DB Server resource.

In the SOAMS-Platform Agent Layer, we use the JMX standard to encapsulate all the resources (hardware and software) with Java objects and expose them in a distributed environment at the resource service point’s side. Sun’s Java Management Extension (JMX) is a new feature in version 5.0 of the Java 2 platform. JMX technology provides the tools for building distributed, Web-based, modular and dynamic solutions for managing and monitoring devices, applications, and service-driven networks [26]. In JMX, Java objects known as Managed Beans, or Mbeans instrument a given resource. These MBeans are registered and managed by management agents, known as JMX agents. The specification provides a set of services for JMX agents to manage MBeans. By design, this standard is suitable for adapting legacy systems, implementing new management and monitoring solutions, and plugging into those of the future.
4.4. Test Experiment and Analysis

The manager subscribes to the overload events from the MUSE-Gateway in advance. When an overload happens, a notification alert message is reported from the web server resource through a JMX mbean, and consequently displayed on the SOAMS-Platform monitor view. The operator takes the operation on the SOAMS-Platform monitor view and executes the scale out operation—adding a new server. And then Web server 3 begins to provide the same web site service as Web server 1 and Web server 2. The load of the target system gradually returns back to normal.

Table II Configuration of JMeter on Each Client PC

<table>
<thead>
<tr>
<th>Client PC</th>
<th>Hardware Description: CPU/Memory/ Harddisk</th>
<th>NumberOf Requests Threshold group*</th>
<th>numberofthreads</th>
<th>Time Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC1</td>
<td>Intel 2.0 GHz/2GB/500GB</td>
<td>5*1600</td>
<td></td>
<td>08:00~08:40</td>
</tr>
<tr>
<td>PC2</td>
<td>Intel 2.0 GHz/2GB/500GB</td>
<td>5*1000</td>
<td></td>
<td>08:10~08:40</td>
</tr>
<tr>
<td>PC3</td>
<td>Intel 1.8 GHz/2GB/500GB</td>
<td>5*600</td>
<td></td>
<td>08:16~08:40</td>
</tr>
<tr>
<td>PC4</td>
<td>Intel 1.8 GHz/2GB/500GB</td>
<td>5*400</td>
<td></td>
<td>08:26~08:40</td>
</tr>
</tbody>
</table>

Fig. 10. Number of ActiveRequests Load of Web Server 1&3

Fig. 10 describes the experiment case in contrast. As shown in Fig. 10, at about 08:26, the overload situation is happening in Web server 1: the number of active requests has reached about 7200. The notification of Web server 1 and 2 overload is received by the operator. The operator then executes the adding A New Server (Web Server 3) operation from the monitor view.

After adding a new Web Server into service pool, Web Server 3 begins to process the users’ requests at about 08:27, as shown in Fig. 10.

The experiment scenario is as follows:

At the beginning, Web Server 1 and 2 run. When the requests increase, the system load goes up.

Table I Description of Target System

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>IP:PortNumber</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Site Service</td>
<td>SearchBook</td>
<td>10.10.82.160:80</td>
<td>CPU/Memory</td>
</tr>
<tr>
<td>Load Balancer</td>
<td>LB</td>
<td>10.10.82.160:80</td>
<td>Intel 2.8GHz/4GB</td>
</tr>
<tr>
<td>Web Server</td>
<td>Web1</td>
<td>10.10.82.161:80</td>
<td>Intel 2.6GHz/2GB</td>
</tr>
<tr>
<td></td>
<td>Web2</td>
<td>10.10.82.162:80</td>
<td>Intel 2.6GHz/2GB</td>
</tr>
<tr>
<td></td>
<td>Web3</td>
<td>10.10.82.163:80</td>
<td>Intel 2.6 GHz/2GB</td>
</tr>
<tr>
<td>Database Server</td>
<td>DB</td>
<td>10.10.82.180:80</td>
<td>Intel 2.8GHz/4GB</td>
</tr>
</tbody>
</table>

Fig. 9. Target Application System

In our current SOAMS-Platform, we realize a web site service management. The target application system has the following characteristics: The target application system is a 3-tiered web-based application system (Web Server/Application Server/Database Server) and the web server tiers can be scaled out, based on the user request pressure.

As described in Fig. 9 and Table I, the target system consists of 5 servers, 1 as the director (load balancer), 3 as the web servers and 1 as database server. The target application system provides one single primitive web site service “SearchBook”. The web server and DB server will provide this service.

As described in Table II, we use four PCs as clients to run the JMeter tool to simulate many requests from end-users. We use LVS as the Load Balancer Server. When a lot of simulated users access LVS, it will direct the users to different web servers according to the load status.

The manager subscribes to the overload events from the MUSE-Gateway in advance. When an overload happens, a notification alert message is reported from the web server resource through a JMX mbean, and consequently displayed on the SOAMS-Platform monitor view. The operator takes the operation on the SOAMS-Platform monitor view and executes the scale out operation—adding a New Server. And then Web server 3 begins to provide the same web site service as Web server 1 and Web server 2. The load of the target system gradually returns back to normal.
There is decline of Web Server 1 starting at 08:27. At the same time, the NumberofActiveRequests of Web server 3 is increasing starting at about 08:27. The overload of Web server 1 is recovered as it returns back to normal. At about 08:40, the NumberofActiveRequests of Web Server 1 is below 6000, and NumberofActiveRequests of Web Server3 is about 5000, as shown in Fig. 10.

5. **WS-MANAGEMENT-BASED SYSTEM RESOURCE MANAGEMENT MIDDLEWARE MODEL AND SCHEME**

5.1. WS-Management Architecture Introduction

As mentioned above, “Web Service for Management” - WS-Management [3] is another standard specification which describes a general Web Services-based protocol for managing systems such as PCs, servers, devices, Web services, other applications and other manageable entities. The WS-Management is published by Distributed Management Task Force (DMTF).

WS-Management addresses the cost and complexity of IT management by providing a common way for systems to access and exchange management information across the entire IT infrastructure. By using Web services to manage IT systems, deployments that support WS-Management will enable IT managers to remotely access devices on their networks - everything from silicon components and handheld devices to PCs, servers and large-scale data centers. WS-Management is the first specification in support of the DMTF initiative to expose CIM resources via a set of Web services protocols.

![Fig. 11.WS-Management related Web Service Standard Specifications Family](image)

Just as Fig. 11 describes, WS-Management related specifications family includes WS-Transfer [31], WS-Eventing [32], WS-Enumeration [33], WS-Addressing [19], and WS-Resource Catalog.

Web Services Addressing (WS-Addressing) [19] specifications have been submitted to W3C as a Member Submission in 2004. By publishing this document, W3C acknowledges that BEA, IBM, Microsoft, SAP, and Sun Microsystems, Inc. have made a formal submission to W3C for discussion.

WS-Addressing provides transport-neutral mechanisms to address Web services and messages. Specifically, this specification defines XML [XML 1.0, XML Namespaces] elements to identify Web service endpoints and to secure end-to-end endpoint identification in messages. This specification enables messaging systems to support message transmission through networks that include processing nodes such as endpoint managers, firewalls and gateways in a transport-neutral manner.

WS-Addressing defines two interoperable constructs that convey information that is typically provided by transport protocols and messaging systems. These constructs normalize this underlying information into a uniform format that can be processed independently of transport or application. The two constructs are endpoint references and message information headers.

WS-Transfer [31] and related specifications has been submitted to W3C as Member Submission in 2006. This specification describes a general SOAP-based protocol for accessing XML representations of Web service-based resources. This specification defines a mechanism for acquiring XML-based representations of entities using the Web service infrastructure. It defines two types of entities: (1) Resources, which are entities addressable by an endpoint reference that provide an XML representation; (2) Resource factories, which are Web services that can create a new resource from an XML representation

WS-Enumeration specifications [33] have also been submitted to W3C as a Member Submission in 2006. This specification describes a general SOAP-based protocol for enumerating a sequence of XML elements that is suitable for traversing logs, message queues, or other linear information models. There are numerous applications for which a simple single-request/single-reply metaphor is insufficient for transferring large data sets over SOAP. Applications that do not fit into this simple paradigm include streaming, traversal, query and enumeration. WS-Enumeration specification defines a simple SOAP-based protocol for enumeration that allows the data source to provide a session abstraction, called an enumeration context, to a consumer that represents a logical cursor through a sequence of data items. The consumer can then request XML element information items using this enumeration context over the span of one or more SOAP messages.

Web Services Eventing (WS-Eventing) [32] has also been submitted to W3C as a Member Submission in 2006. This specification describes a protocol that allows Web services to subscribe to or accept subscriptions for event notification messages. Web services often want to receive messages when events occur in other services and applications. A mechanism for registering interest is needed because the set of Web services interested in receiving such messages is often unknown in advance or will change over time. This specification defines a protocol for one Web service (called a “sub-
scriber") to register interest (called a "subscription") with another Web service (called an "event source") in receiving messages about events (called "notifications" or "event messages"). The subscriber may manage the subscription by interacting with a Web service (called the "subscription manager") designated by the event source.

To improve robustness, a subscription may be leased by an event source to a subscriber, and the subscription expires over time. The subscription manager provides the ability for the subscriber to renew or cancel the subscription before it expires. There are many mechanisms by which event sources may deliver events to event sinks. This specification provides an extensible way for subscribers to identify the delivery mechanism they prefer. While asynchronous, pushed delivery is defined here, the intent is that there should be no limitation or restriction on the delivery mechanisms capable of being supported by this specification.


Based on WS-Management, we propose a novel system resource middleware model; the whole model can be divided logically into three layers: the manager layer, the middleware layer and the manageable resource layer. The architecture is demonstrated in Fig. 12 (WS-Management based System Resource Management Middleware Model):

Manageable Resource Layer: includes all kinds of managed IT resources, like computers, routers, databases app servers, operation systems etc. The type of managed IT resources depends on the management model. In this layer, all the managed IT resources provide specific manageability interfaces (that can be JMX [26], WMI [27], SNMP [28] or even private protocols, etc.).

Middleware Layer: every managed IT resource has its corresponding standard management model in this layer. It provides WS-Management standardized manageability interfaces for SRMAs to manipulate the managed IT resources. Logically, every management model (also known as web service resource type) represents a managed IT resource type. Every management model can have many instances. Each instance is one kind of physical managed IT resource.

Therefore, we can build all kinds of standardized WS-Management resource models in the middleware layer to manage many actual IT resources in the resource layer- even though they are physically distributed in different environments. For example, we design a CPU management model to represent the CPUs in the PC resource. When we build such a management middleware model, we can then use web service manageability interfaces to manipulate all the actual CPUs that are configured to be the management model’s instances.

Management Layer: contains all the management policies and logic, and uses the client side manageability interfaces of web service resource types in the middleware layer to manipulate the managed IT resources. These client side manageability interfaces correspond to the server side of the standard manageability interfaces in the middleware layer. Because all resource management models conform to the same standards in the middleware layer, the Management Layer can easily manage all the managed IT resources in a standardized way.

5.3. Wiseman and WMI Based System Resource Management Scheme Implementation

Based on the WS-Management Middleware Model, we propose and implement a Wiseman and WMI based resource management scheme, as shown in Fig. 13. Wiseman [4] is a open-source Java implementation of WS-Management; it can be freely adopted in the enterprise management software. The project scope includes the WS-Management specification and its dependent specifications. This framework shows us a way to use Wiseman in a real system resource management scenario. It has three layers: Resource Agent Layer, Gateway Layer, and Manager Layer. The manager layer is the abstract layer where system management software is located. The manager layer demonstrates an overview of how WS-Management Resources are accessed by a remote system manager. They are the consumers of WS-Management-based manageability interfaces that are exposed by the Wiseman server in the Gateway Layer.

In Gateway layer, we use Wiseman server to contain WS-Management resources. In this layer, WS-Management compliant manageability interfaces are exposed and the system manager can access these resources through WS-Management interface.
The Resource Agent Layer is the abstract layer where existing resource manageability interfaces are exposed; resource manageability protocol can be SNMP, JMX, WBEM or any specific manageability protocols. In our implementation scheme environment, we use WMI (an implementation of WBEM) in the Agent Layer.

The reasons we chose WMI is because Windows Management Instrumentation (WMI) [27] is the Microsoft implementation of Web-based Enterprise Management (WBEM) [29], which is an industry initiative to develop a standard technology for accessing management information in an enterprise environment. WMI uses the Common Information Model (CIM) [25] industry standard to represent systems, applications, networks, devices, and other managed components. The ability to obtain management data from remote computers is what makes WMI useful. WMI is supported by Windows platforms which are the popular platforms in PCs and Servers. In order for Wiseman server to visit WMI resource, we use COM4J library. COM4J is A Java library that allows Java applications to seamlessly interoperate with Microsoft Component Object Model [35].

5.4. Test Case of Wiseman Eventing Mechanism
5.4.1 Wiseman Eventing Mechanism Design Architecture

Here we design an application of an eventing mechanism using a Wiseman system in our test case. By connecting to the WMI Event Mechanism, we setup a scenario of using the Wiseman library in the real system resource management scenario and test the functionality of the Eventing Mechanism in the Wiseman library.

In our test case, an application of Wiseman Eventing Mechanism is deployed in a single machine that acts as a gateway. Through standard interfaces (WS-Management compliant interfaces), we subscribe or unsubscribe to the Wiseman resources located in the gateway, and receive the WS-Management compliant notifications sent by these Wiseman resources. WS-Management compliant notifications are transformed from WMI Notifications that are triggered by the WMI Event Mechanism in the lower layer in real time. The overall architecture of our design is demonstrated in the following Fig. 14 Wiseman Eventing Mechanism Design in our Test Case.

Eventing applications consist of three sub-parts: “Subscribe”, “Notify” and “Unsubscribe”. In our test case, subscribe requests are carried out in the Gateway Layer and the Resource Agent Layer. The notifications are first generated by the WMI Event Mechanism in the Agent Layer, and then translated to WS-Management compliant events. Then, these events are sent out to their consumers through the Gateway Layer.

The detail of notifications is demonstrated in the following Figure-Fig. 15. Sequence Diagram of Eventing Notification:

Fig.13. Wiseman and WMI Based System Resource Management Scheme

Fig.14. Wiseman Eventing Mechanism Design in our Test Case

Fig.15. Sequence Description of Eventing Notification
In our test case, unsubscribe requests are carried out in both the Gateway Layer and the WMI Agent Layer.

### 5.4.2 Data Structure of the Eventing Test case

There are two fundamental data structures: a resource schema and a specific filter type used to subscribe and filter notifications. Here we describe Filter Type: Gauge Filter, as shown in List 1. It is a developer-defined filter type used to subscribe and filter notifications. Using this kind of filter, clients can start specific subscriptions with respect to max-threshold and min-threshold of some attributes’ value.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"

<xsd:complexType name="GaugeFilterType">
  <xsd:sequence>
    <xsd:element name="Name" type="xsd:string"/>
    <xsd:element name="MaxThreshold" type="xsd:float"/>
    <xsd:element name="MinThreshold" type="xsd:float"/>
  </xsd:sequence>
</xsd:complexType>
```

**List 1. Gauge Filter**

### 5.4.3 Test Environment of Wiseman Eventing Mechanism

The test environment of a Wiseman eventing mechanism is shown in Fig. 16. The module for a client is packed in SubscribeClient.jar. The module for a Wiseman server is packed in WS-CPU-Resource.war. The module of Gateway calls the module of Com4j.dll to connect WMI resources.

![Fig.16. Test Environment of Wiseman Eventing Mechanism](image)

### 5.4.4 Test Case Executing of Wiseman Eventing Mechanism

Subscribing and Monitoring: CPU Threshold. First we carry out subscription in the SubscribeClient. After the CPU value is bigger than the subscribed threshold, the Target Managed Resource will send a notification to Wiseman’s server side; the events will be printed on the console.

![Fig.17. Memory Usage in Scenario 1](image)

1. SubscribeClient Interface. For example, subscribing to resource 10.10.82.153, the threshold is 50% (0.5);
2. The Wiseman Server side Interface of receiving a notification.

The process of the notification is to print the events and obtain the CPU-id of the Target Managed Resource that has sent the notification.

**Test Scenario:** Two management clients executes “Subscribe”, “Notify” and “Unsubscribe” on different resource instances.

**Test Objective:** In order to observe whether the system management operator can successfully subscribe to multiple resources events and receive notifications. At the same time, we observe the memory usage in the Wiseman Server.

**Test Type:** function test & performance test

**Test for User Type:** Operator

**Test Client Number:** 2

**Managed Resource Number:** 18

**Multiple Scenarios:**

**Scenario 1:** Issue “subscribe” requests from two clients to many resource instances.

**Test Objective:** Subscribe many instances from two clients.

**Test Type:** function test & performance test

**Actual Result:** we can successfully subscribe every instance on multiple resources

**The memory usage in the Wiseman server is:** Normal; with the increasing of subscriptions, memory usage increases from 8MB to 15 MB around, as shown in Fig17.

**Scenario 2:** Notification from multiple resource instances that are subscribed by two clients.

**Test Objective:** Subscribe from two clients to many resource instances. The two event sink will receive multiple events respectively.

**Test Type:** function test & performance test

35
**Actual Result:** We can see many notifications in the management console.

**The memory usage in the Wiseman server:** We can see the memory usage increase sharply (from 8MB to 16MB) when notification sending in parallel, as shown in Fig. 18.

### 5.5. Test Case of Wiseman Transfer and Enumeration Mechanism

#### 5.5.1 Test Environment of Wiseman Transfer and Enumeration Mechanism

The implementation environment of Wiseman Transfer and Enumeration Mechanism is shown in Fig. 19:

The client module is packed in WisemanTE-Client.jar. The server module is packed in WisemanTE-OS.war. The module of Gateway is packed in WMI_Process.jar. The WisemanTE-Client will call the WisemanTE-OS interface through WS-Management protocol to carry out management operation, and the WisemanTE-OS will further call WMI COM interface through the Gateway-WMI_Process.jar to execute the corresponding resource management operation.

![Test Environment of Wiseman Transfer and Enumeration Mechanism](image)

**Fig. 19. Test Environment of Wiseman Transfer and Enumeration Mechanism**

The interfaces provided by Gateway are shown in list 2:

```
//Function 1: return the all processes on a special IP
public WMI_Process[] getProcesses(String IP, String Uid, String Pwd)
```

```
//Function 2: return the process information of a special process on a special IP
public WMI_Process getProcessInfo(String...
```

### 5.5.2 Test Case Executing of Wiseman Transfer and Enumeration Mechanism

#### Scenario 1:

Get the top 20 processes on a special PC. Top 20 is the 20 processes with the highest memory usage value from No.1 to No.20.

**Test Objective:** Test the Enumerate(filter), Pull, Release operations in WS-Enumeration.

**Test Type:** function test

**Test for User Type:** Operator

**Test Client Number:** 1

**Managed Resource Number:** more than 540 (18 PC x 30 processes)

**Test Sequence of Each Client:** Enumerate(filter) -> Pull - > ... -> Release

**Description:**

This test case is to consummate the functionality of the...

![Test Result of Wiseman Transfer and Enumeration Mechanism](image)

**Fig. 21. Test Result of Wiseman Transfer and Enumeration Mechanism**

---

*List 2: the Gateway Management Interface*

---

IP, String Uid, String Pwd, int ProcessId)

//Function 3: terminate a special process on a special IP
public void terminateProcess(String IP, String Uid, String Pwd, int ProcessId)

//Function 4: set the new priority for a special process on a special PC
public void setPriority(String IP, String Uid, String Pwd, int ProcessId, int newPriority)

---

**Fig. 18. Memory Usage in Scenario 2**

**Fig. 20. Memory Consumption Snapshot of JProbe in Scenario 1**
eventing mechanism of Wiseman in a realistic usage.
When testing, the special PC is randomly selected from all
the managed PCs.
Top 20 is the 20 processes with the highest Memory usage
value from No.1 to No.20.
Why use top-20 processes as the parameter? Why not more?
Because too many processes are not needed to return to the
client. 20 is enough. (20/500=4%)
**Input Parameter:** Grinder 3.0 is used. The Python script is
executed. The following is the snapshot of the Grinder3.0.
**Desired Output:** Return the properties of a special process on a special PC.

**Fig. 23. Memory Consumption Snapshot of JProbe in Scenario 2**

**Actual Result:**
Fig. 22 lists the properties of a special process on a special
PC. Fig. 23 is the snapshot of JProbe, which displays the
memory consumption of Tomcat. When the enumerate operation is executed, the memory consumption rises
from 14MB to 22MB. The pull operation takes a little time. When the release is executed, the memory consumption
decreases. The Get operation is not reflected in the Fig.23.

**5.6. Evaluation of Test Case**

We have tested the following Wiseman and WS-Management Elements: Transfer and Enumeration, Subscribe, Sink/Event Notify, Unsubscribe; all the above functionality items have been verified.

Through these tests on the Wiseman software, we have confirmed that Wiseman can finish the same tasks (Transfer and Enumeration, Subscribe, Sink/(Notify), Unsubscribe) in the System Management Domain.

Understanding problems or difficulties of using Wiseman and WS-Management is very important. Through this test case, we give the following evaluation consideration:

1) We found that when we carry out multiple subscribe and unsubscribe requests, some exceptions appear, but we cannot yet identify whether it is a Wiseman’s problem or a com4J’s problem. Therefore, we will test further.

2) One difficulty is how to inform Wiseman clients of the specific filters' usage or filtering conditions.

3) The Eventing Mechanism of Wiseman makes the event sources loosely coupled with their event sinks. To be able to get events, the event sink requires no predefined interfaces. Sending and receiving events are all through a general HTTP connection. The only question is how event sinks know schemas of received events and retrieve their contents.

4) Wiseman provides high reusability of its core code. On the server side, while building middleware with Wiseman, developers can easily add custom capabilities and user-defined filters into their middleware structure without affecting the core code. And without changing Wiseman’s core code, developers can use Wiseman to adapt current existing management interfaces (such as WMI, JMX, SNMP, and so

WS-Management is one specification focusing on the system management technology field, which can not satisfy all the requirements of system management in some aspects. We refer to these limitations as gaps. The first gap is between the WS-Management specification and the latest Wiseman software[4], which is a Java-based implementation of WS-Management. The second kind of gap is between the WS-Management specification and the real system management requirement. We have analyzed the first gaps in [37]. Here, we analyze the second kind of gap and design some possible solutions for some typical gaps.

5.7.1 Knowing the Number of Events at Event Source in Pull Mode

A. Description

In some circumstances, polling for events is an effective way of controlling data flow and balancing timeliness against processing ability. Also, in some cases, network restrictions prevent “push” modes from being used; that is, the service cannot initiate a connection to the subscriber.

However, there are some demerits in this mode when it is used in the subscription. If pull mode is active and a wsm:Pull request returns no events (because none have occurred since the last "pull"), the service should return a wsm:TimedOut fault.(Line3312-Line3313 in WS-Man specification [3]).

On one hand, this method is necessary. On the other hand, clients do not know when the best time to pull the events is. This is because the client doesn’t know the number of events which are contained in the event source.

Sometimes clients will pull too often but only get few events, or even no event. This is a waste of network bandwidth.

Sometimes clients will pull too few so that a lot of events have to be kept at the event source. This is a waste of memory at event source.

Furthermore, the event sink may be inconvenient, because the event sink does not know how to decide the interval between the pull operations.

B. Analysis

However, this problem doesn’t exist in Enumeration. To give clients an estimate of the number of items in an enumeration, two optional SOAP headers are defined: one for use in the request message to return an approximate count of items in an enumeration sequence, and a corresponding header for use in the response to return this value to the client. This mechanism is intended to assist clients in determining the percentage of completion of an enumeration as it progresses.

Therefore, we can add a mechanism in pull mode to help event sink to know more about the situation of event source.

C. Solution

- We add two new elements in the pull response
  - “RestEventsCount”: To indicate how many events still remain at the event source
  - “EstimatedOptimalInterval”: To indicate an estimation of the best interval between two pull operations calculated by event source, e.g. the event source can estimate an interval using the average speed of event-happening.

![Fig.24. Scenario 1](image)

As Figure24 Scenario1 shows at the beginning, the interval between pull operations is 4 seconds and at most 2 events can be delivered at once. But every 1 second, the queue in event source can add a new event, so the queue will become longer and longer. The solution is that the event sink can shorten the interval to 3 seconds.

As Fig. 25 Scenario 2 shows, at the beginning, the interval between pull operations is 1 second and 2 events can be delivered once at most. But every 2 seconds the queue in the event source can add a new event, so the pull response is always faults. The solution is that event sink can prolong the interval to 2 seconds. At the same time, the client receives too many 0’s and the number of the events is fewer than 2, so client can prolong the interval to 3 seconds.
D. Conclusion
This gap item has been proposed as specifications to China Standards body.

5.7.2 The Sampling Interval of Subscriptions
A. Description
In the real management scenario, management software mainly takes the active mode to monitor resources. The management software will get the performance data from the resource every fixed interval (e.g. 5 minutes).

If the management software exposed the WS-Man interface, how can we implement it?

B. Solution
Choice 1: there is a timer on the side of the client. When the time is up to 5 minutes, the client will send the “Get” request to the service side, then the service will return the performance data to the client side, as Figure 26 shows.

Choice 2: we can subscribe to the resource. In addition, the services will deliver the event (performance data) to the client. We can set the low threshold (e.g. 0) to ensure that the event must be emitted, as shown in Figure 27. We know that there are 4 kinds of parameters needed in this scenario. They are: starting time, ending time, the sampling interval and the monitored resource.

The starting time: the time of the subscription built up.

The ending time: the expiration time of the subscription.

The sampling interval: is not defined in the WS-Man.

The monitored resource: the Resource URI of subscription request.

Each resource model needs a different sampling interval. The client needs to designate it in the “subscribe” request. Such as:

```
<ws:Delivery>
  ...
  <ws: SamplingInterval> 5 minutes </ws: SamplingInterval>
  ...
</ws:Delivery>
```

The following Table III compares these above two solution choices.

<table>
<thead>
<tr>
<th>Choice</th>
<th>Advantage</th>
<th>Disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Get” the performance data every fixed time (e.g. 5 minutes)</td>
<td>As the “get” operation is one-off operation and the service doesn’t need to maintain something.</td>
<td>Increase the message processing of client and service. Increase the network traffic.</td>
</tr>
<tr>
<td>“Subscribe” to the resource to get the event(performance data) every fixed time(e.g. 5 minutes)</td>
<td>Client side only needs to send request once. Decrease the message processing of client and service. Decrease the network traffic.</td>
<td>Subscription Manager needs to maintain the subscription.</td>
</tr>
</tbody>
</table>

C. Conclusion
This gap item has been proposed as specifications to China Standards body. We can add a new element to designate the sampling interval in the subscribe request for monitoring the performance data.
5.7.3 Heartbeat Can Be Independent from Subscription

A. Description
The purpose of heartbeats is described as follows:

- The heartbeat event is sent if no regular events occur so that the client knows the subscription is still alive.
- If the heartbeat event does not arrive, the client knows that connection is bad OR that the subscription has been invalid because of failure or expiration, and it can take corrective action then. (Line 2839-Line 2845 in WS-Man specification).

![Fig.28. A common architecture: Subscription-specific Heartbeat](image)

To request heartbeat events as part of a subscription, the wse:Subscribe request has an additional field in the wse:Delivery section:

```
<wse:Delivery>
...<wsman:Heartbeats>xs:duration</wsman:Heartbeats>
...
</wse:Delivery>
```

Usually, as Figure28 described, there are many subscriptions on one WS-Man service. The WS-Man service has to respectively send a heartbeat event for each subscription. A lot of bandwidth is consumed with this technique.

Furthermore, even if such subscription-specific heartbeat is applied, it is impossible for a client to discriminate the heartbeat between the two following reasons when no event (regular event or heartbeat event) is received:
- The network connection is bad
- The subscription has become invalid

B. Solution and Analysis
The solution is that the heartbeat is independent of the subscription, as shown in Figure29. Table IV compares the diagnostic functions of two kinds of heartbeats.

If the heartbeat independent of subscription is implemented, the SubscriptionEnd MUST also be implemented. The following Table IV compares the heartbeat independent of subscription with the subscription-specific heartbeat. We can see the advantage of the heartbeat independent of subscription.

![Fig.29. A new architecture: the heartbeat independent of subscription](image)

<table>
<thead>
<tr>
<th></th>
<th>Subscription-specific heartbeat</th>
<th>Heartbeat independent of subscription</th>
</tr>
</thead>
<tbody>
<tr>
<td>When a client receives heartbeat events</td>
<td>The client can deduce that (1) the network connection is ok AND (2) the subscription is valid</td>
<td>The client can deduce that (1) the network connection is ok AND (2) One subscription is valid if the client receives no SubscriptionEnd message relating to it. (3) One subscription is invalid if the client receives one SubscriptionEnd message relating to it.</td>
</tr>
<tr>
<td>When a client receives no heartbeat events</td>
<td>It is impossible for the client to discriminate between the two reasons: (1) the network connection is bad OR (2) The subscription is invalid. So line 2872–2874 in WS-Man writes: The client will assume that the subscription is no longer active if no heartbeats are received within the specified interval, so the service can proceed to cancel the subscription and send any requested SubscriptionEnd messages, as the client will likely re-subscribe shortly.</td>
<td>The client can only deduce that (1) The network connection is bad. (2) And this is not contrary to the left rules in WS-Man, i.e. The client will assume that all the subscriptions on a certain WS-Man service are no longer active if no heartbeats are received within the specified interval, so the service can proceed to cancel all the subscriptions and send any requested SubscriptionEnd messages, as the client will likely re-subscribe shortly.</td>
</tr>
</tbody>
</table>


6. WSDM and ws-MANAGEMENT STANDARD STACK comparison

In this paper, we discuss how Web Service architecture can be used in the system resource management field. Moreover, we mainly focus on the Web Services management interfaces mapping mechanism. We also test functionality and performance of WSDM and WS-Management. We do not focus on Web Services standard based management system performance comparison. We note Moura, G. C. M. [34] et al. have compared the performance of two Web services-based management specifications: WSDM-MUWS and WS-Management - against SNMP. The experiments related to network usage have shown that management Web services compliant to either MUWS or WS-Management are much more verbose than those used in previous investigations (9:1 and 16:1 in relation to SNMP). Although there is some room to lower network usage (e.g., by choosing shorter attribute names), the size of SOAP request and response messages will still be much larger than SNMP messages, given the huge differences of size observed.

The paper [34] also described the response time comparisons: WS-Management has performed about 38% worse than SNMP, while WSDM-MUWS presented half the performance of SNMP. Although they have not evaluated the amount of memory consumed, they have observed that the WS-Management agent has allocated 35 MB of RAM memory at startup, while the WSDM-MUWS agent has allocated 52 MB. Most of this memory is used by the Java Virtual Machine, Tomcat, and the agent. Putting WSDM-MUWS and WS-Management in perspective, WS-Management presented the best results for the three metrics investigated: network usage, response time, and CPU usage. When we test Wiseman Transfer and Enumeration Mechanism, Wiseman memory consumption is between 4MB to 30MB. These results reflect the nature of WS-Management, which is informally declared to be more lightweight than WSDM-MUWS. Although WS-Management imposes less bandwidth than WSDM-MUWS, it is still very verbose. That may be a major concern if it is considered to operate in restricted bandwidth environments; fortunately, such environments are becoming rare today.

Here, we further compare WSDM and WS-Management standard stack from perspective of function in the following table:

<table>
<thead>
<tr>
<th>Table V Comparison between WSDM and WS-Management Standard Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standard Comparison</strong></td>
</tr>
<tr>
<td><strong>Standard Organization And Current version</strong></td>
</tr>
<tr>
<td><strong>Resource Definition &amp; Status Retrieve</strong></td>
</tr>
<tr>
<td><strong>Event and Notification</strong></td>
</tr>
<tr>
<td><strong>Resource Addressing</strong></td>
</tr>
<tr>
<td><strong>Basic Standard</strong></td>
</tr>
<tr>
<td><strong>Future Update and Extension</strong></td>
</tr>
</tbody>
</table>

Table V compares WSDM and WS-Management’s standard stack. For Resource Definition & Status Retrieve aspect, we can see WSDM using OASIS-WSRF, and WS-Management based on W3C: WS-Enumeration and WS-Transfer. Web Services Resource Framework (WSRF) is a family of OASIS-published specifications for web services. Major contributors include the Globus Alliance and IBM. A web service by itself is nominally stateless, i.e.WSRF provides a set of operations that web services may implement to become stateful; web service clients communicate with resource services which allow data to be stored and retrieved.

For Event and Notification aspect, WSDM utilizes OASIS-WS-Notification, while WS-Management uses W3C-WS-Eventing. For Resource Addressing aspect, both WSDM and WS-Management are based on W3C: WS-Addressing. At the same time, they both utilize basic Standard: W3C:XML, SOAP; and IETF:HTTP, TCP.

As Table V shows, when we compare WSDM and WS-Management’s future prospects, we found WSDM specification has not updated since 2006, the WS-Management have more support from big IT manufacturers, such as Microsoft, Intel, and VMware. At the same time WS-Management has entered into the ISO standard series in 2012. DMTF WS-Management working group is considering simplifying itself using Restful web service and JSON rather than SOAP/XML-based Web Services. Target version 2.0 will be released at 2015 Q2.
7. Conclusion and Future Work

In this paper, we discuss how SOA architecture can be used in the system resource management field. Based on WSDM standard specification, we design a novel Web Service-based management middleware model to support SOA-based SNMAs. At the same time, based on WS-Management specification, we design a novel Web Service-based management middleware model to support SRMAs.

Specifically, our main contribution is that we consider how Web Services management standards are mapped and integrated with other management interfaces, such as JMX, WMI, through middleware model. Based on the WSDM open source implementation-MUSE, we have developed and implemented a MUSE and JMX based resource management scheme. Based on the WS-Management open source implementation-Wiseman, we have developed and implemented a Wiseman and WMI based resource management scheme. Through middleware gateway, we transform the specific manageability interfaces to web service standard based manageability interfaces, and achieve the object of universal management and interoperability of distributed IT resources. At the same time, we also test and verify these schemes' function and performance. We also analyze the gap between WS-Management and real management field and design some feasible solutions.

In our current Web Service-based management middleware models, we only realize the management case of multiple resources or a single service. Actually, we don’t consider the complex composite service. In general, composing multiple Web services, rather than accessing a single service, is essential and provides more benefits to users. Composition primarily addresses the situation of a user’s request that cannot be satisfied by any available service, whereas a composite service obtained by combining available services might be used. When the complex Web Services composition is used in the Web Service-based management model, the following problem must be solved:

The structure of the composite services, which is described by BPEL, or some other language, is complex. If the management platform plans to manage the composite service, the structure of the management middleware model, especially the Gateway layer, might need to be re-designed. And the target service might need to re-build. We might need to consider the metrics of the composite service.

In our scheme, we use JMX and WMI as sample resources to connect into a WSDM-Gateway and a WS-Management-Gateway. In real management scenarios, there exist still a lot of heterogeneous system resources, which expose different local management interfaces. Therefore, the next area that needs further study is smooth mapping mechanisms from these local interfaces to web service management interfaces. In other situations, some resources can directly provide the Web Service manageability interface. Microsoft has implemented the WS-Management standard in Windows Remote Management 1.1 (WinRM), available for Windows XP, Windows Server 2003, and Windows Server 2008. WinRM 3.0 for Windows 7 and Windows Server 2008 R2 was released on Sept 4 2012 and shipped in Windows 8 and Windows Server 2012. What makes WS-Management so useful is the fact that it rides on HTTP/HTTPS and packages its data in XML or JSON packets. We should consider extending our model to manage this kind of resource.

In the near future, there are two technology trends in system resource management applications; one is Web Services-based management model, such as WS-Management. Another is cloud resource management standard. Cloud computing and virtualization technologies are rapidly being adopted by enterprise IT managers to better deliver services to their customers, lower IT costs and improve operational efficiencies. DMTF’s Cloud Management Initiative is focused on developing interoperable cloud infrastructure management standards and promoting adoption of those standards in the industry. DMTF’s Cloud Management Working Group (CMWG) has developed Cloud Infrastructure Management Interface (CIMI) specification. OASIS has built Cloud-Specific or Extended Technical Committees (TC) to focus on cloud standard work, such as OASIS Cloud Application Management for Platforms (CAMP) TC.

8. Acknowledgement

This paper is based on the Fudan-Hitachi Innovative Software Technology Joint Laboratory (InTech) project- Standard-based Distributed System Management Architecture. The authors wish to thank Hitachi researchers for all the support and advice for the Joint Laboratory. This work is also supported by 2014-2016 PuJiang Program of Shanghai under Grant No. 14PJ1431100.

9. References


DMTF - Web-Based Enterprise Management (WBEM)-, http://www.dmtf.org/standards/wbem/.


Authors

Zhihui Lu is an Associate Professor in School of Computer Science, Fudan University. His research interests are service computing, system resource management, and network multimedia technology. He is an expert member of China SOA and cloud standard work group led by CESI. He is also an academic alliance member of DMTF WS-Management Work Group.

Jie Wu is a Professor at School of Computer Science, Fudan University. His research interests are service computing, cloud computing, and system resource management technology. Professor Wu is one expert member of WSSG (Web Service Study Group) and SOA-SG(SOA Study Group) of of ISO/IEC JTC1 SC7, SC38 on behalf of China.

Patrick Hung is an Associate Professor at the Faculty of Business and Information Technology in University of Ontario Institute of Technology. Patrick has been working with Boeing Research and Technology in Seattle, Washington on aviation services-related research projects. He owns a U.S. patent on Mobile Network Dynamic Workflow Exception Handling System with Boeing. His research interests include services computing, business process and security.