[MS-WMSO]:
Windows Management Services System Overview

This document provides an overview of the Windows Management Services System Overview Protocol Family. It is intended for use in conjunction with the Microsoft Protocol Technical Documents, publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts. It assumes that the reader is either familiar with the aforementioned material or has immediate access to it.

A Protocol Family System Document does not require the use of Microsoft programming tools or programming environments in order to implement the Protocols in the System. Developers who have access to Microsoft programming tools and environments are free to take advantage of them.

Abstract

Windows management systems are designed to allow a user to monitor, troubleshoot, and conduct hardware and software inventories of remote computers.

This document describes the intended functionality of the Windows Management System, how it interacts with applications that need computer management, and how it interacts with management clients that need to configure and manage the system. The Windows Management System supports multiple protocols for computer management. This document lists those supported protocols and how they interact in a combined system.

Revision Summary

Date	Revision History	Revision Class	Comments
04/10/2009	0.1	Major	First Release.
05/22/2009	1.0	Major	Updated and revised the technical content.
07/02/2009	1.1	Minor	Updated the technical content.
08/14/2009	2.0	Major	Updated and revised the technical content.
09/25/2009	2.1	Minor	Updated the technical content.
11/06/2009	2.2	Minor	Updated the technical content.
12/18/2009	3.0	Major	Updated and revised the technical content.
01/29/2010	3.0.1	Editorial	Revised and edited the technical content.
03/12/2010	3.0.2	Editorial	Revised and edited the technical content.
04/23/2010	3.0.3	Editorial	Revised and edited the technical content.
06/04/2010	3.0.4	Editorial	Revised and edited the technical content.
07/16/2010	3.0.4	No change	No changes to the meaning, language, or formatting of the technical content.
08/27/2010	3.0.4	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	4.0	Major	Significantly changed the technical content.

Date	Revision History	Revision Class	Comments
11/19/2010	5.0	Major	Significantly changed the technical content.
01/07/2011	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
03/25/2011	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
05/06/2011	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
06/17/2011	5.1	Minor	Clarified the meaning of the technical content.
09/23/2011	5.1	No change	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	5.1	No change	No changes to the meaning, language, or formatting of the technical content.
03/30/2012	5.1	No change	No changes to the meaning, language, or formatting of the technical content.
07/12/2012	5.1	No change	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	5.1	No change	No changes to the meaning, language, or formatting of the technical content.
01/31/2013	5.1	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

This Protocol Family System Document (PFSD) is primarily intended to cover the Protocol Family as a whole. In conjunction with Member Protocol Technical Documents (TDs), which are intended to cover Member Protocols, it presents the rules for information exchange relevant to those Member Protocols and the Protocol Family that are used to interoperate or communicate with a Windows operating system in its various environments.

Asset management, monitoring of system health and diagnosis of computer failure in large-scale distributed systems requires a common mechanism for representing, retrieving, and manipulating data about system operations and health.

The Windows Management System provides a unified object model and network protocols for remote inspection and modification of system management data.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

access control list (ACL)
Common Information Model (CIM)
Common Information Model (CIM) namespace
Common Information Model (CIM) object
Common Information Model (CIM) Object Manager (CIMOM)
credential
Distributed Component Object Model (DCOM)
encoding
Hypertext Transfer Protocol (HTTP)
role
security descriptor
SOAP
URI
Windows Management Instrumentation (WMI)
WMI Query Language (WQL)

The following terms are specific to this document:

Action URI: A **URI** that identifies which operation needs to be carried out against a **resource**.

asset: Computers, hardware, and so on, in the Inventory Management sense.

DMTF: The Distributed Management Task Force.

ResourceURI: A **URI** that identifies resources that are used by management services that implement a Windows Management System (**WMS**) protocol. A ResourceURI can be used to access objects of **Common Information Model (CIM)** and **Windows Management Instrumentation (WMI)** classes.

Security Descriptor Description Language (SDDL): A text-based format for specifying **security descriptors** ([MS-DTYP] section 2.5.1).

WMI: The Windows Management Instrumentation Remote Protocol [MS-WMI]. This is the protocol that implements **WMI**.

WMS: The Windows Management System.

WMS application: An application that uses **WMS** to monitor, set up, configure, troubleshoot, or inventory some set of remote computers.

WSMAN: The Web Services Management Protocol Extensions for Windows Server 2003 [MS-WSMAN].

WSMV: The Web Services Management Protocol Extensions for Windows Vista [MS-WSMV].

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. Note that in [RFC2119] terms, most of these specifications should be imperative, to ensure interoperability. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

Any specification that does not explicitly use one of these terms is mandatory, exactly as if it used MUST.

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[DMTF-DSP0004] Distributed Management Task Force, "Common Information Model (CIM) Infrastructure Specification", version 2.3, October 2005, http://www.dmtf.org/standards/published_documents/DSP0004V2.3_final.pdf

[DMTF-DSP0226] Distributed Management Task Force, Inc., "Web Services for Management (WS-Management) Specification", version 1.0.0, February 2008, http://dmtf.org/sites/default/files/standards/documents/DSP0226_1.0.0.pdf

[DMTF-DSP0227] Distributed Management Task Force, Inc., "WS-Management CIM Binding Specification", version 1.0.0, June 2009, http://www.dmtf.org/sites/default/files/standards/documents/DSP0227_1.0.0.pdf

[MS-AUTHSO] Microsoft Corporation, "Windows Authentication Services System Overview".

[MS-DCOM] Microsoft Corporation, "Distributed Component Object Model (DCOM) Remote Protocol".

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

[MS-KILE] Microsoft Corporation, "Kerberos Protocol Extensions".

[MS-NLMP] Microsoft Corporation, "NT LAN Manager (NTLM) Authentication Protocol".

[MS-WMI] Microsoft Corporation, "Windows Management Instrumentation Remote Protocol".

[MS-WMIO] Microsoft Corporation, "Windows Management Instrumentation Encoding Version 1.0 Protocol".

[MS-WSMAN] Microsoft Corporation, "Web Services Management Protocol Extensions for Windows Server 2003".

[MS-WSMV] Microsoft Corporation, "Web Services Management Protocol Extensions for Windows Vista".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.org/rfc/rfc2119.txt

[RFC2371] Lyon, J., Evans, K., and Klein, J., "Transaction Internet Protocol Version 3.0", RFC 2371, July 1998, http://www.ietf.org/rfc/rfc2371.txt

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt

[SOAP1.2-1/2003] Gudgin, M., Hadley, M., Mendelsohn, N., et al., "SOAP Version 1.2 Part 1: Messaging Framework", W3C Recommendation, June 2003, http://www.w3.org/TR/2003/REC-soap12-part1-20030624

[SOAP1.2-2/2003] Gudgin, M., Hadley, M., Mendelsohn, N., et al., "SOAP Version 1.2 Part 2: Adjuncts", W3C Recommendation, June 2003, http://www.w3.org/TR/2003/REC-soap12-part2-20030624

1.2.2 Informative References

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary".

2 Overview

Introduction (section 1) describes this Protocol Family System Document. This section introduces the system that is being documented.

2.1 System Summary

The **Windows Management System (WMS)** provides an infrastructure that enables a user or application to inspect, modify, and monitor an **asset's** resources remotely.

In WMS, resources are represented as objects, following the **Common Information Model (CIM)**.

WMS includes three independent protocols used to access **CIM objects**. A particular implementation of WMS might support one or more of these three protocols, as described in section 4.5.

2.2 List of Member Protocols

The Windows Management System can include one or more of the following member protocols:

Windows Management Instrumentation Remote Protocol [MS-WMI]: This protocol allows reading and writing of CIM objects across multiple computers. This is a **Distributed Component Object Model (DCOM)**-based protocol.

Web Services Management Protocol Extensions for Windows Server 2003 [MS-WSMAN]: This protocol allows reading and writing of CIM objects across multiple computers. This protocol is based on a prerelease draft version of WS-Management, and is incompatible with current **DMTF** specifications.

Web Services Management Protocol Extensions for Windows Vista [MS-WSMV]: This protocol allows reading and writing of CIM objects across multiple computers. This protocol is based on version 1.0 of WS-Management.

Each of the WMS protocols provides access to the **CIM Object Manager (CIMOM)** resources of hosts within the system, but they differ in certain aspects. Some major differences in capability are as follows:

- Windows Management Instrumentation Remote Protocol (**WMI**): Smaller messages than **WSMV** and **WSMAN** due to use of binary message **encoding** instead of **SOAP**. See [MS-WMI] section 2.2 for details of message encoding. This protocol offers methods to modify the CIM repository on a managed host.
- Web Services Management Protocol Extensions for Windows Server 2003 operating system: **HTTP**-based protocol allows for easier network configuration than WMI when **WMS applications** and managed computers may be separated by a firewall. This protocol is based on a prerelease draft of the WS-Management specification, and is not compatible with the final 1.0 version.
- Web Services Management Protocol Extensions for Windows Vista operating system: HTTP-based protocol allows for easier network configuration than WMI when WMS applications and managed computers may be separated by a firewall. The protocol offers methods to control remote command-line shells on a managed host. See [MS-WSMV] section 3.1.4.1.31 for details.

In addition to these protocols, WMS includes the following data structure:

- Windows Management Instrumentation Encoding Version 1.0 [MS-WMIO]: This data structure specifies a binary data encoding format that is used by the Windows Management Instrumentation Remote Protocol for network communication.

2.3 Relevant Standards

The Windows Management System depends on the following systems and protocols:

- **Distributed Management Task Force WS-Management** protocols described in [DMTF-DSP0004], [DMTF-DSP0226], and [DMTF-DSP0227]. These protocols detail the encoding of CIM data, and define the abstract object model.
- **Kerberos Protocol Extensions** specified in [MS-KILE] and **NT LAN Manager (NTLM) Authentication** protocol specified in [MS-NLMP]. These protocols authenticate identities and group them. The system restricts access to its assets and functionality to specified groups.
- **DCOM Remote Protocol**, as specified in [MS-DCOM]. WMI uses DCOM for transmission of data between WMS client and server, and for authentication of client and server.
- **SOAP**, as specified in [SOAP1.2-1/2003]. This standard defines message format and semantics used by WSMV and WSMAN.
- **DNS**, as specified in [RFC2371]. This standard is used for locating monitored computers.
- **TCP/IP**, as specified in [RFC2371]. This standard is used for data transmission for the underlying network.
- **HTTP/1.1**, as specified in [RFC2616]. This standard is used for data transmission by Web Services Management Protocol Extensions for Windows Server 2003 operating system, as specified in [MS-WSMAN], and by Web Services Management Protocol Extensions for Windows Vista operating system, as specified in [MS-WSMV].

3 Foundation

This section describes the theoretical and practical information needed to understand this document and this system.

3.1 Background Knowledge and System-Specific Concepts

With constant advances in the capability, scalability, and affordability of computing and communications technology, there are a few noticeable trends in the way corporations manage their operations:

- The number of computers in the workplace that are used to accomplish day-to-day tasks is greatly increasing.
- The diversity of computers in the workplace is increasing, and now includes desktop computers, laptops, servers, and mobile devices.
- More organizations are opening branch offices in remote locations, and those branch offices still need access to the data and computing resources of the central office.
- More organizations are using data center services provided by specialized companies.

As a result of these trends, the job of managing a company's IT infrastructure is becoming a both a complicated and mission-critical task. An administrator needs to be able to monitor computers and software, collect and analyze performance data, and carry out actions while rarely having direct physical access to the computers themselves. For example, an IT administrator may need to simultaneously manage the power consumption of servers in a data center, the security settings for the operating systems running on office desktops, and the configuration options for specific applications the employees use to get work done.

Another major complicating factor is the diversity of the computers that need to be managed. This diversity manifests itself in a number of ways, including the following:

- Different categories of computers: desktops, laptops, servers, mobile phones, and more.
- Computers run on different processors – for instance, 32-bit chips or 64-bit chips.
- Computers have different operating systems and applications installed.

In order to simplify the management of a heterogeneous and widely-distributed collection of computers, it is necessary to provide both a common mechanism for retrieving and manipulating data, and a common format for representing that data. In this way, a single management application or interface can be used across the entire organization, meaning the IT administrator will know how to work with all of the computers, and any new computer that is added to the organization will be compatible with the existing management applications and tools. This consistent representation needs to be shared by all device manufacturers in order for it to be effective, so it has to be able to support a variety of devices with different capabilities.

Consistent data representation is accomplished by the CIM standard ([DMTF-DSP0004]), which has been ratified by the DMTF. CIM is a conceptual model that is not bound to any particular implementation. It also allows for vendor extensions, so that it is not too restrictive to be useful. Because of this, any system that exposes CIM-compliant data will be able to be accessed in a consistent manner, and if a particular vendor requires additional functionality they can extend the standard CIM schema.

WMS is the Microsoft software that exposes CIM data in Windows. The method through which this data is provided can be broken down into three logical components:

- The Common Information Model (CIM) Object Manager (CIMOM), which serves as the interface through which CIM objects are accessed.
- The CIM Repository, which stores the metadata that defines the available CIM classes.
- The CIM managed objects, which represent the actual data.

WMS defines a way by which management applications and tools can access CIM data remotely. The data can be retrieved through one of the three independent remote protocols included in WMS:

- The Windows Management Instrumentation Remote Protocol, specified in [MS-WMI].
- The Web Services Management Protocol Extensions for Windows Server 2003 operating system, specified in [MS-WSMAN].
- The Web Services Management Protocol Extensions for Windows Vista operating system, specified in [MS-WSMV].

The three remote protocols act as a communication between the management application and the CIMOM. The management application issues a request to the client role of the member protocol, which sends the request to the server role of the member protocol, which then delivers the request to the CIMOM. Upon receipt of a request, the CIMOM retrieves the available CIM classes and their schema from the CIM Repository, retrieves the actual instances of CIM objects, and exposes them to the server role of the member protocol. This data is then delivered back to the client role of the member protocol and then to the management application.

These protocols do not interact with one another – the only way in which they are related is that they can all be independently used to access the same CIM data. The protocols each have various properties and capabilities that differentiate them, and an application determines which protocol to implement.

Familiarity with the CIM object model and the general functionality of a CIMOM is required to understand this system. For a description of the theoretical functionality of a CIMOM, see [MS-WMI] section 3.1.4.3, which contains a subset of the operations that define the functionality of a CIMOM.

The reader should also be familiar with basic network-security concepts such as authentication, message integrity, and encryption. It is not necessary to understand the details of a specific security mechanism.

3.2 System Purposes

The WMS System is used primarily for four general categories of tasks:

- Inventory of hardware and software assets
- Setting up, configuring, and updating assets
- Monitoring asset status
- Troubleshooting problems

A given WMS application can accomplish these tasks by using the protocols in WMS System to communicate with one or more managed computers. A single WMS application may use more than one WMS protocol to accomplish its goals. Also, some managed computers may not support all three

of the WMS member protocols, so the WMS applications may need to use different WMS member protocols to accomplish the same task on different managed computers. Alternatively, a WMS application may use a particular member protocol in order to leverage additional features in the protocol that are beyond the scope of the WMS System (for example, remote shell capabilities). Differences in protocol capabilities are discussed in section 5.3.1.

A further source of complexity is that a single organization may use multiple WMS applications to control different aspects of the same set of computers, and the WMS application may use different protocols to accomplish the same task on a single managed computer.

Each of the management tasks described in this section corresponds to a series of one or more operations on CIM objects available through some set of computers. The specification for each protocol in the WMS System describes the set of supported operations for that protocol. From a system standpoint, the only cross-protocol interaction is the fact that each application communicating with a given managed host is operating on the same CIM data, regardless of the protocol chosen by the application.

Therefore, when a WMS application reads from a host's CIMOM, such as by enumerating instances of a class or querying properties of an instance, the system is required to provide a consistent view of the CIM database of objects, regardless of the protocol used for the query. Similarly, when a WMS application modifies CIM state, all future observers MUST see the same change.

3.3 System Use Cases

This section describes the summary-level and user-level use cases for the Windows Management Services System.

3.3.1 Stakeholders and Interests Summary

The stakeholders are WMS users, the Windows Management Server, the Windows Management Client, architects, and developers.

Windows Management Server: This stakeholder implements a CIMOM and responds to network requests that are initiated by the Windows Management Client.

WMS User: This stakeholder can be a person or management application. The user needs to access or change some set of CIM data for some set of computers. The primary interest of the WMS user is to manage a set of computers.

There are four key categories of tasks that this stakeholder carries out:

- Taking inventory of assets
- Configuring assets
- Monitoring assets
- Troubleshooting problems

Windows Management Client: This stakeholder is responsible for providing an interface to the CIM data for some set of computers. The primary interest of the Windows Management Client is to provide such functionality.

Architect: This stakeholder designs systems that can be monitored for failures and corrected properly. The WMS System offers an architect a common paradigm for remote access to many types of host resources using the industry-standard CIM model. This simplifies the design, implementation, and testing of the resulting system.

To access CIM resources, WMS offers the architect a set of network protocols with different protocol dependencies and relationships to public standards. The system offers a consistent view of the underlying CIM resources, regardless of which protocol is used to access the resources.

Developer: This stakeholder implements the WMS System or subcomponents of the WMS that will interoperate with other WMS systems. The developer's primary interest is in creating the tools necessary to access and manipulate CIM data in a way that is interoperable with existing WMS systems.

3.3.2 Supporting Actors and System Interests Summary

Authentication System: The Authentication System as specified in [MS-AUTHSO] provides authentication services through NTLM, Kerberos, Basic, and Digest authentication to secure communications in the Windows Management System and the authentication services that support the client to server communication within and outside the Windows Management System.

3.3.3 Use Case Diagrams

The use cases discussed in detail in section 3.3.4 are grouped into four high level scenarios described in this section.

Use case group	Use cases
Asset Management	Query Properties of a CIM Instance – Windows Management Client
Setup, Configuration, and Update	Create a CIM Instance – Windows Management Client Invoke a Method on a CIM Instance – Windows Management Client Set Properties of an Instance – Windows Management Client
Monitoring	Query Properties of a CIM Instance – Windows Management Client Monitor Events from WMS – Windows Management Client
Diagnosis and Troubleshooting	Set Properties of an Instance – Windows Management Client Query Properties of a CIM Instance – Windows Management Client

3.3.3.1 Asset Management

In order to manage a system, a WMS User needs to first collect information about the computers or devices available to be managed. For example, a WMS User may need to know per-computer information such as MAC and IP addresses, hardware ID numbers, firmware or software version numbers, or processor architecture. It is important to note that this inventory cannot just be taken once. Due to computer failures, device purchases, and the prevalence of laptops and mobile devices, the collection of devices is dynamic. So, the information needs to be able to be collected remotely without relying on static, pre-populated data.

Managing assets is a one-to-many operation with one WMS application querying the state of many computers, such as servers in a data center or desktops in a business, and providing IT personnel a consolidated view of the results. The console issues requests to enumerate CIM resources and query their properties.

The following diagram illustrates the use cases for asset management.

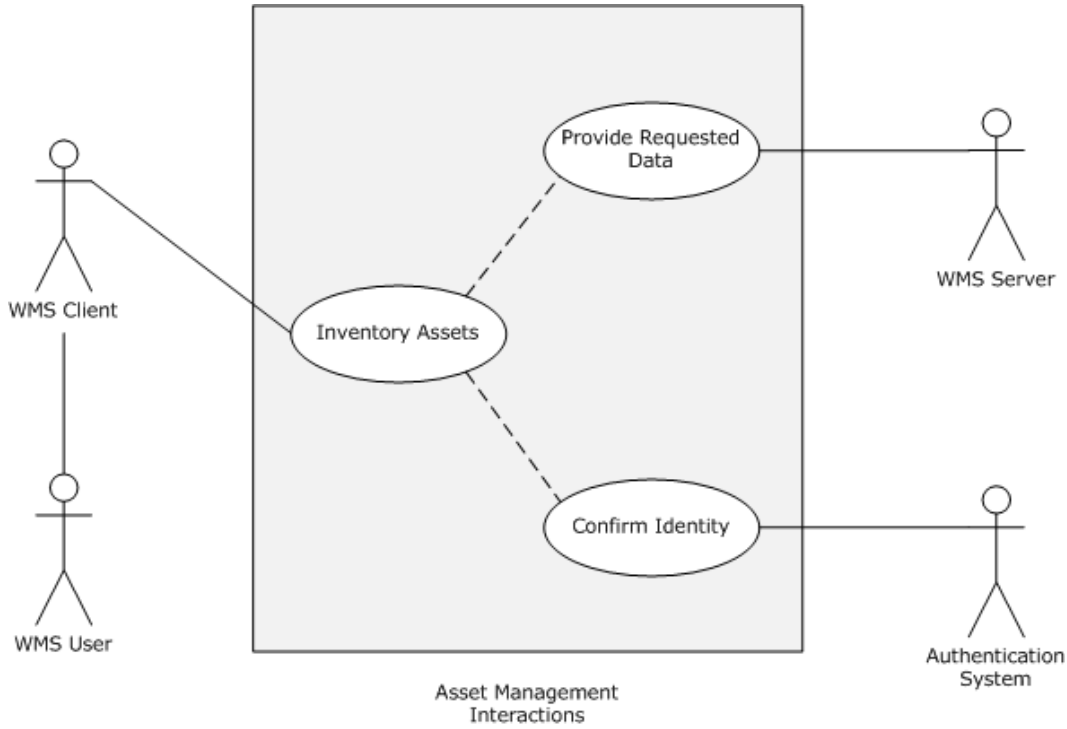

Figure 1: Asset Management use cases diagram

3.3.3.2 Setup, Configuration, and Update

An IT administrator needs to be able to set up and configure the computers for use. Once a computer becomes a part of the corporate network, it needs to be updated with the required software applications, operating system security patches, and any other settings needed to comply with corporate policies. This configuration is not a one-time operation; as software updates and updated security mechanisms such as virus definitions become available, or as network and IT policies change, an IT administrator needs to be able to run the necessary updates on all affected computers on the network.

Setting up, configuring, and updating management data is a one-to-one or one-to-many operation with one WMS application changing the state of one or many managed entities. The console issues requests to create CIM resources and modify their properties. These activities often rely on the data gathered by the asset management and monitoring use cases.

The following figure depicts a summary-level diagram of the use cases described in section 3.3.4.

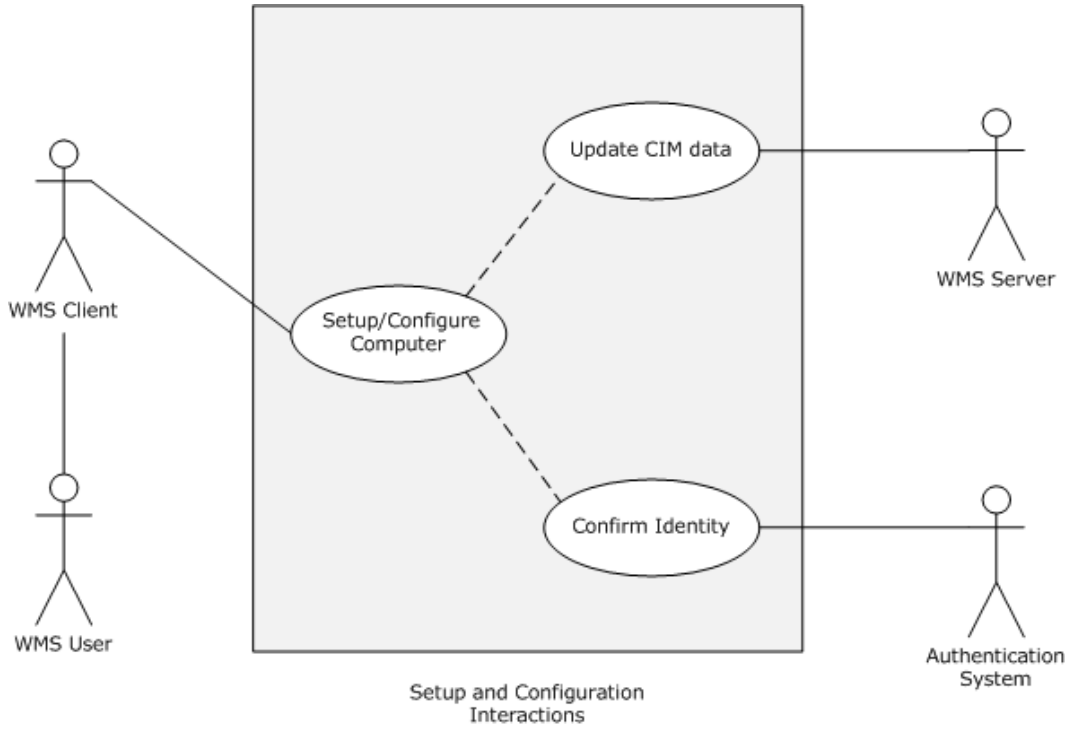

Figure 2: Setup, Configuration, and Update use cases diagram

3.3.3.3 Monitoring

Once all of a company's assets have been configured and connected to a network, an IT administrator needs to monitor the computers and their status. This is crucial for a number of reasons: to gather performance data and assess the efficiency of the company resources, to identify security breaches caused by malicious applications or unauthorized computers, to discover computer failures or other technical problems, to audit activity for legal reasons, and many other activities. This monitoring inherently collects dynamic data, either in real time or after storage and aggregation on the local computers.

Monitoring is a one-to-many operation with one WMS application and many managed entities. The WMS application may subscribe to updates of the managed computer's CIM resources, or periodically query the state of CIM resources, or both.

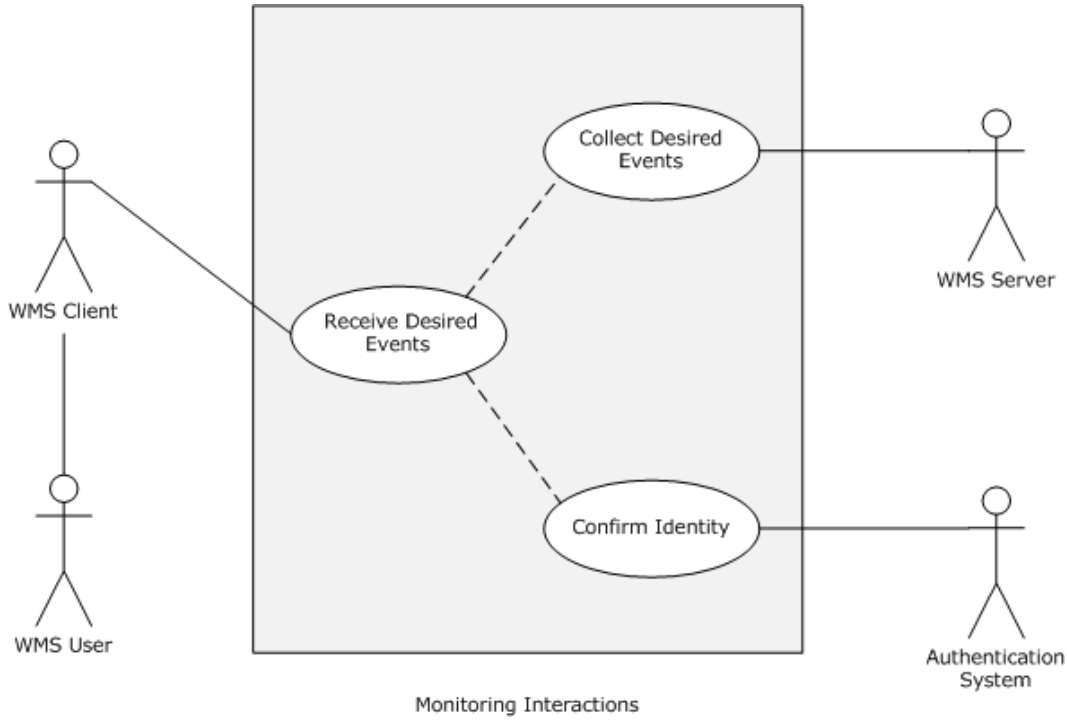

Figure 3: Monitoring use cases diagram

3.3.3.4 Diagnosis and Troubleshooting

When an IT administrator is alerted that a failure has occurred or is soon to occur, it is not always possible to directly diagnose the computer in question – this is often done remotely. The remedy may be carried out programmatically based on data collected through monitoring, or the IT administrator may be alerted to the problem and explicitly connect to and diagnose the problem computer. Because some of the more severe failures can leave a computer's operating system in a non-responsive state, it is often necessary to carry out troubleshooting and diagnostics by directly communicating with a computer’s hardware.

Diagnosis is a one-to-one operation where the WMS application or IT administrator queries and sets the state of CIM objects on a single host in order to diagnose and correct problems.

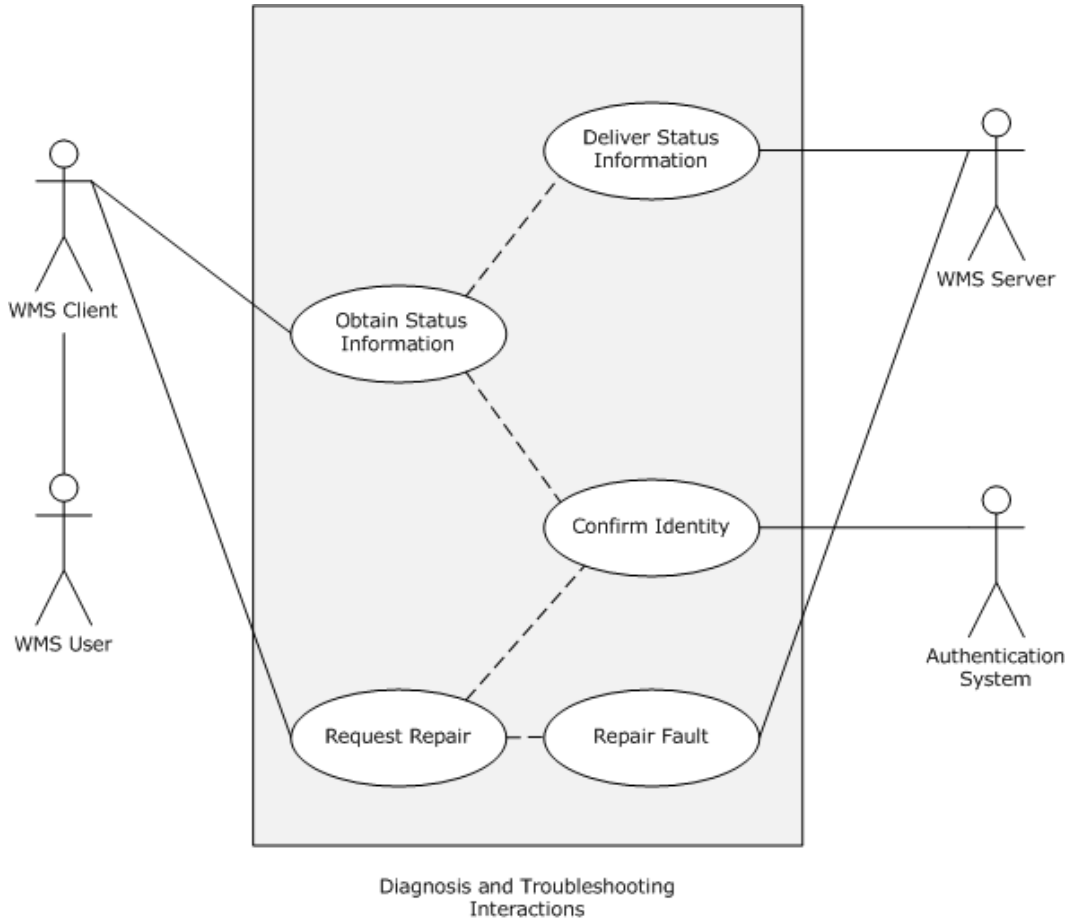

Figure 4: Diagnosis and Troubleshooting use cases diagram

3.3.4 Use Case Descriptions

The following use case descriptions detail the types of functionality available through WMS.

3.3.4.1 Create a CIM Instance — Windows Management Client

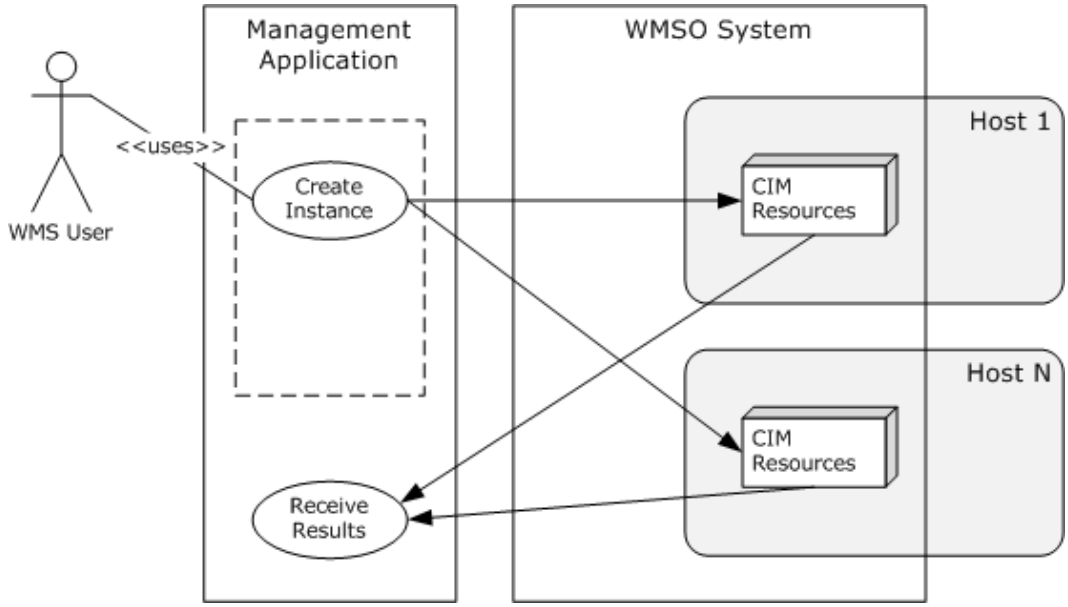

Figure 5: Creation of a CIM instance

Goal: To use the **Windows Management Client** to create a new CIM object in the CIM repository. The newly-created object will have attributes populated as specified by the client.

Context of Use: This use case typically occurs when a WMS User wishes to set up or configure a computer for the first time, or when the WMS User wishes to install a new software application.

Direct Actor: The direct actor in this use case is the Windows Management Client.

Primary Actor: The primary actor in this use case is the WMS User.

Supporting Actors: The supporting actor is the Authentication System, as specified in [MS-AUTHSO].

Stakeholders and Interests:

- The WMS User, for creating the instance.

Preconditions:

- There MUST NOT already exist a CIM object with the same name as the CIM object which is to be created.
- For creation of a class, the WMS User needs to know the intended schema.
- For creation of an instance, the WMS User needs to know the intended class, its schema, the intended name of the instance, and the intended values of instance properties.

Minimal Guarantees: The CIM repository is not corrupted.

Success Guarantee: The WMS System guarantees that the desired instance was created and populated correctly.

Trigger: This use case is triggered by a request from the WMS User.

Main Success Scenario:

1. The **Windows Management Client** sends a request to the server specifying the class name and instance name of the CIM object to be created.
2. The WMS creates a CIM object with the name and class supplied by the client.
3. The server notifies the client that the object was created successfully.

Extensions: None.

3.3.4.2 Invoke a Method on a CIM Instance — Windows Management Client

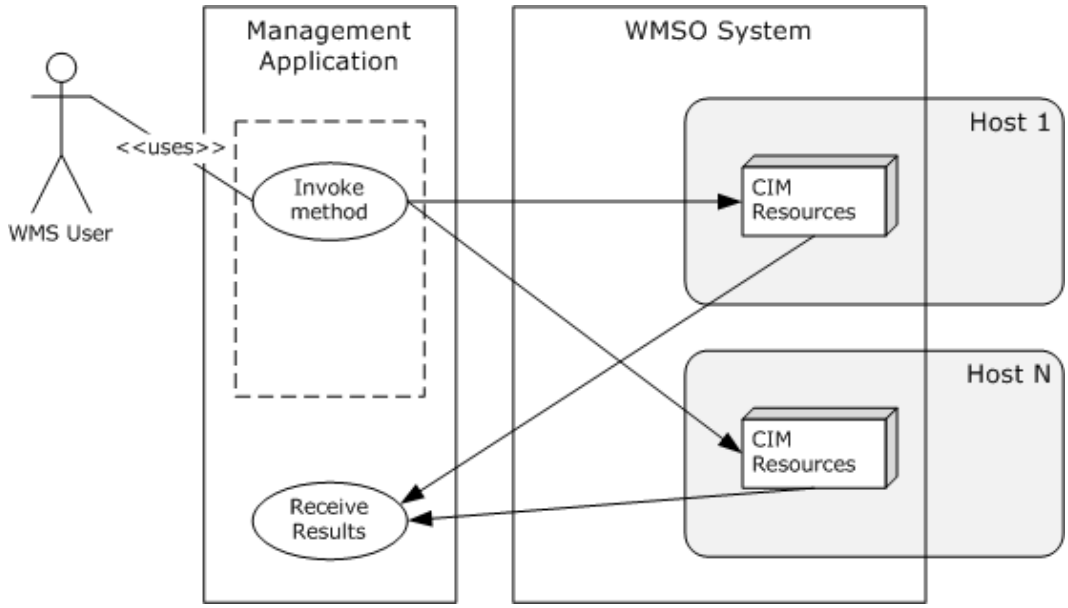

Figure 6: Invoking a method on a CIM instance

Goal: To invoke a method on a managed object.

Context of Use: This use case typically occurs when a WMS User wishes to take an action on a managed object which is supported by the CIM methods on that object.

Direct Actor: The direct actor is the Windows Management Client.

Primary Actor: The primary actor is the WMS User.

Supporting Actors: The supporting actor is the Authentication System, as specified in [MS-AUTHSO].

Stakeholders and Interests: The WMS User, for achieving the desired action.

Preconditions:

- The WMS User needs to have a reference to a specific CIM instance or class.

- The WMS User needs to know the name of the method to be called, as well as the name and type of the method parameters.

Minimal Guarantee: There are no minimal guarantees.

Success Guarantee: The WMS System guarantees that the method was executed, and the results are returned to the WMS User.

Trigger:

- This use case is triggered by a command from the WMS user. The WMS System then takes action.

Main Success Scenario:

- The **Windows Management Client** sends a request to the server specifying the name and class of the CIM object to be accessed and the name of the method to be executed on that object.
- The server verifies that the targeted CIM object implements the specified method.
- The server executes the method.
- The server notifies the client that the method was invoked successfully.

Extensions: None.

3.3.4.3 Set Properties of an Instance — Windows Management Client

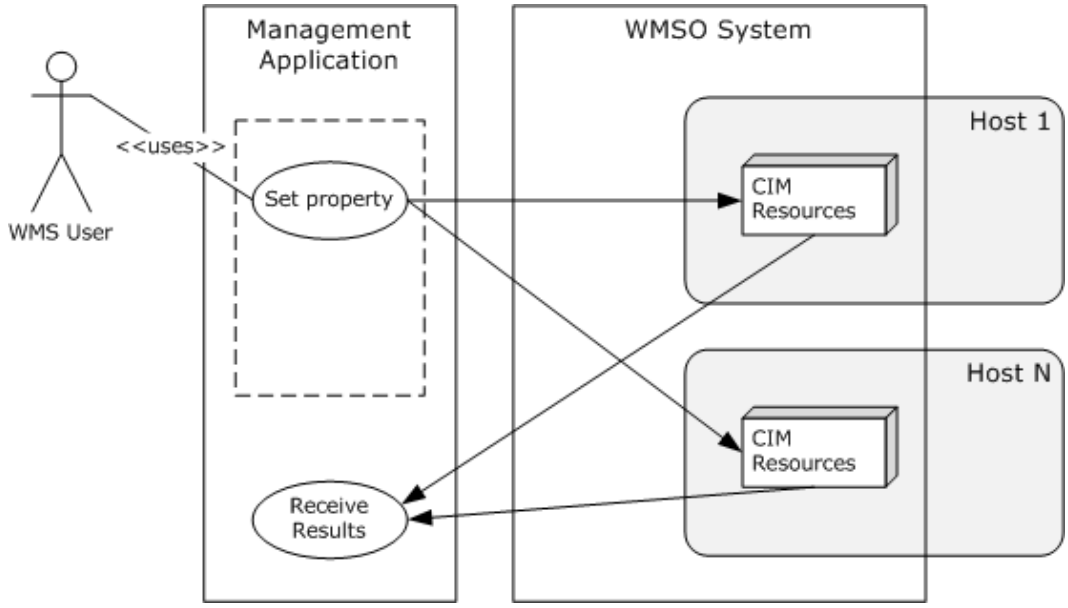

Figure 7: Setting properties of an instance

Goal: To set one or more properties of an existing CIM instance.

Context of Use: This use case occurs when the WMS User wishes to change some properties of a CIM instance.

Direct Actor: The direct actor is the Windows Management Client.

Primary Actor: The primary actor is the WMS User.

Supporting Actors: The supporting actor is the Authentication System, as specified in [MS-AUTHSO].

Stakeholders and Interests: The WMS User, to set the desired properties.

Preconditions:

- The WMS User needs to have a reference to a specific CIM instance or class.
- The WMS User needs to know the data type for each property that is being modified.
- The WMS User needs to know the intended value of each property that is being modified.

Minimal Guarantees: Each individual property is either set to the requested value, or it is not modified from the current value.

Success Guarantee: The WMS System guarantees that the properties were correctly set.

Trigger:

- This use case is triggered by a command from the WMS user.

Main Success Scenario:

1. The **Windows Management Client** sends a request to the server specifying the name of the CIM class, instance name, and set of properties that need to be modified..
2. The WMS modifies the instance with the data supplied by the client. No other property of the instance is modified other than explicitly stated by the client
3. The Server notifies the client that the requested properties for the CIM instance were modified successfully.

Extensions: None.

3.3.4.4 Query Properties of a CIM Instance — Windows Management Client

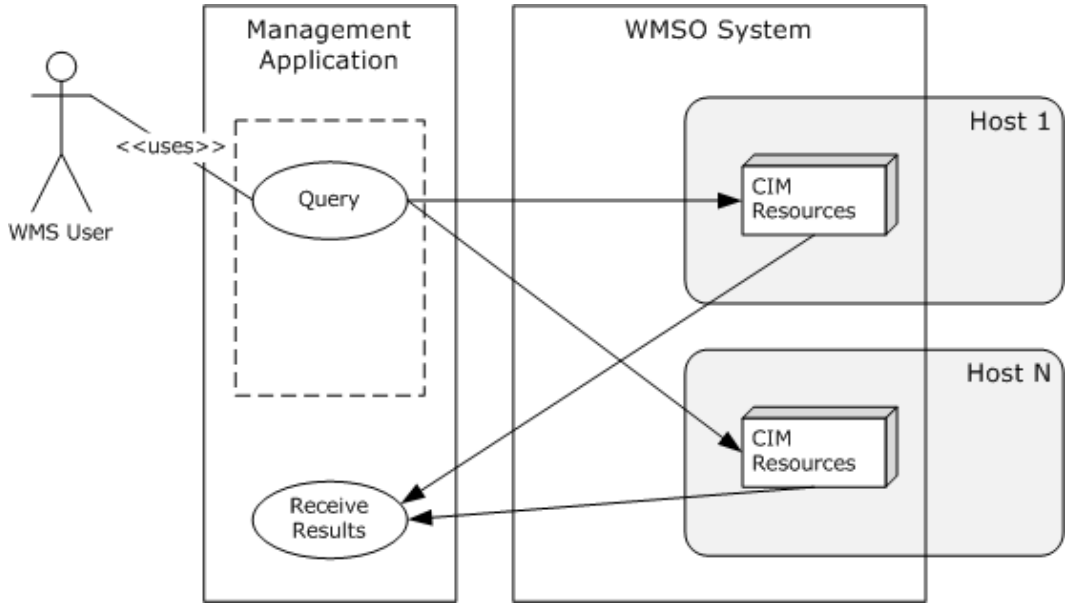

Figure 8: Querying properties of a CIM instance

Goal: To retrieve the values of some set of properties of a CIM instance.

Context of Use: This use case occurs whenever the WMS User wishes to determine the current status of some managed object.

Direct Actor: The direct actor is the Windows Management Client.

Primary Actor: The primary actor is the WMS User.

Supporting Actors: The supporting actor is the Authentication System, as specified in [MS-AUTHSO].

Stakeholders and Interests: The WMS User, for retrieving the requested data.

Preconditions:

- The queried properties have to exist.
- The WMS User needs to know the class of the instance.
- The WMS User needs to know the schema of the instance.
- The WMS User needs to know the instance-name of the instance.

Minimal Guarantee: There are no minimal guarantees.

Success Guarantees: The WMS System guarantees that the requested data was retrieved.

Trigger: This use case is triggered by a request from the WMS User.

Main Success Scenario:

1. The client sends a request to the server specifying the name of the CIM class, instance name, and set of properties being queried.

2. The WMS queries for the properties of the particular instance of the CIM class. This is a read-only operation.

3. The server successfully returns the values of the requested properties for the CIM instance to the client.

Extensions: None.

3.3.4.5 Monitor Events from WMS — Windows Management Client

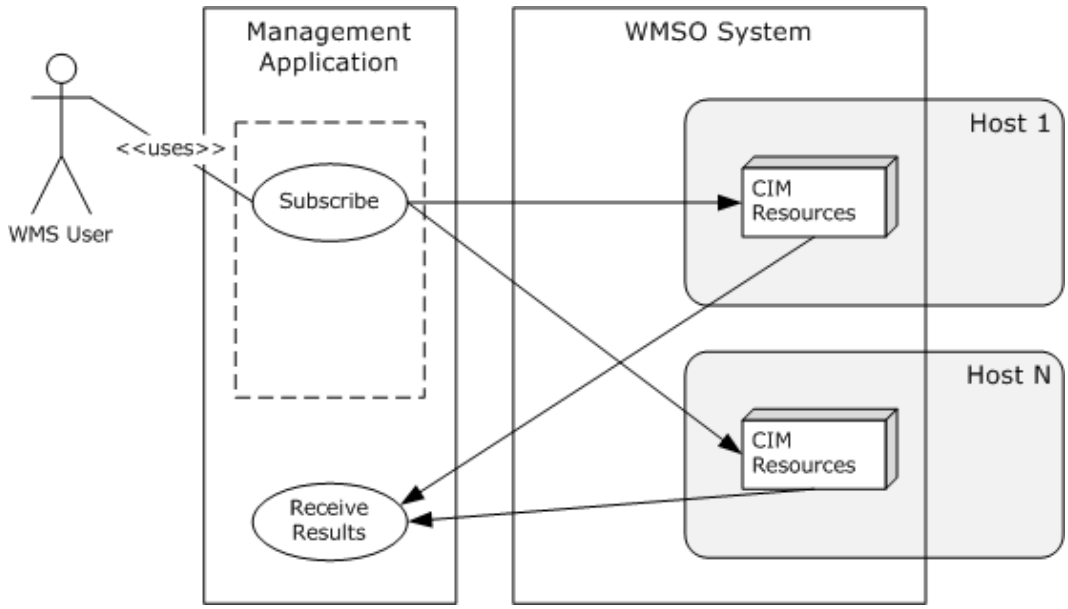

Figure 9: Monitoring events from WMS

Goal: The goal is to submit a query to WMS and to have WMS to notify the submitting user when conditions given in the query are met.

Context of Use: This use case occurs when the WMS User wishes to be alerted to specific changes in the monitored objects.

Direct Actor: The direct actor is the Windows Management Client.

Primary Actor: The primary actor is the WMS User.

Supporting Actors: The supporting actor is the Authentication System, as specified in [MS-AUTHSO].

Stakeholders and Interests: The WMS User, to monitor the desired events.

Preconditions:

- The **Windows Management Client** knows the type of events to which it will subscribe.

Minimal Guarantees: There are no minimal guarantees.

Success Guarantees: The WMS System guarantees to deliver all requested notifications of events to the WMS User.

Trigger:

- This use case is triggered by a command from the WMS user. The WMS System then stores the query and executes it based on query conditions.

Main Success Scenario:

1. **Windows Management Client** sends query to the server.
2. If or when the server generates events matching the query, then the server delivers event notifications to the client.

3.3.4.6 Delete CIM Object — Windows Management Client

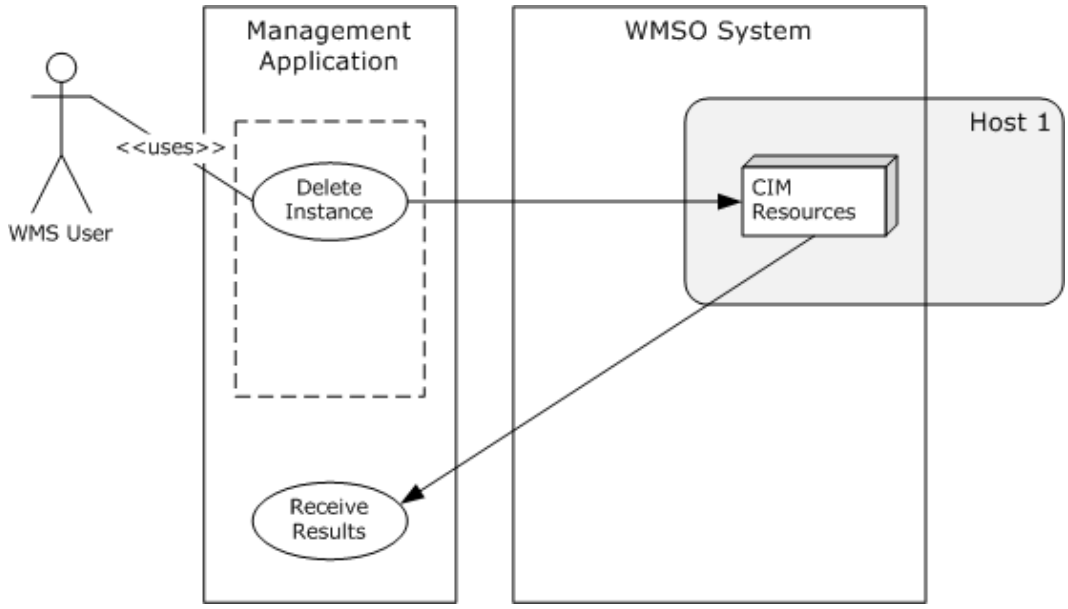

Figure 10: Deleting CIM object

Goal: To use the **Windows Management Client** to delete an existing CIM object in the CIM repository.

Context of Use: This use case typically occurs when a WMS User wishes to remove a previously created object such as an environment variable.

Direct Actor: The direct actor in this use case is the Windows Management Client.

Primary Actor: The primary actor in this use case is the WMS User.

Supporting Actors: The supporting actor is the Authentication System, as specified in [MS-AUTHSO].

Stakeholders and Interests:

- The WMS User, for deleting the instance.

Preconditions:

- A CIM object MUST already exist with the same class name and instance name as the CIM object which is to be deleted.

Minimal Guarantees: There are no minimal guarantees.

Success Guarantee: The WMS System guarantees that the desired instance was deleted.

Trigger: This use case is triggered by a request from the WMS User.

Main Success Scenario:

1. The **Windows Management Client** sends a request to the server specifying the name and class of the CIM object to be deleted.
2. The WMS deletes the CIM object with the name and class supplied by the client.
3. The Server notifies the client that the object was deleted successfully.

Extensions: None.

3.3.4.7 Attempt Delete of CIM Object — Windows Management Client

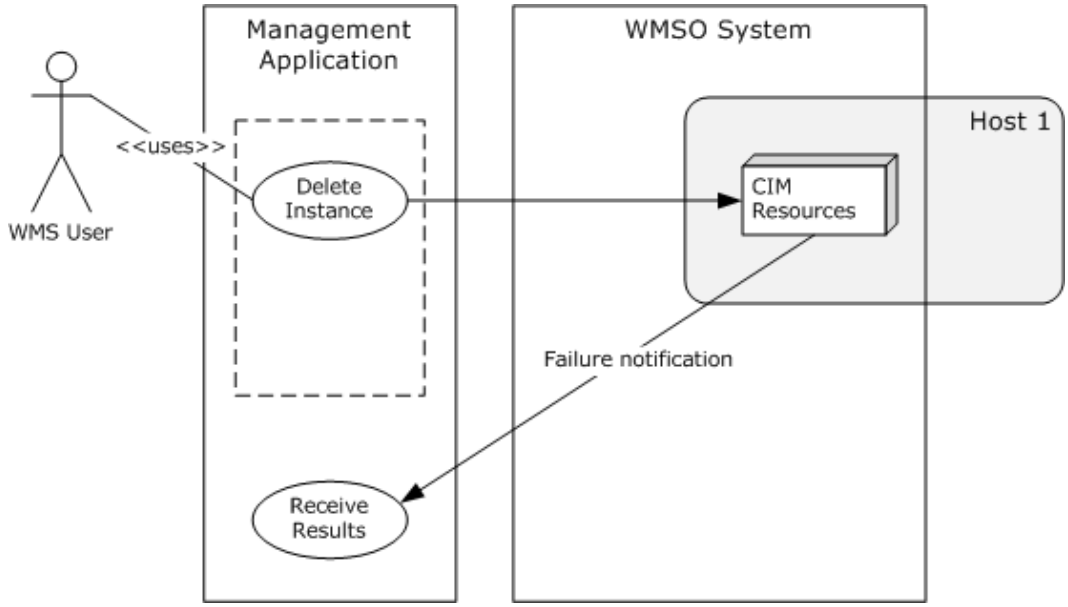

Figure 11: Attempted delete resulting in failure

Goal: To illustrate the failure of a requested CIM object deletion.

Context of Use: This use case occurs when the WMS User requests deletion of a non-existent CIM object.

Direct Actor: The direct actor is the Windows Management Client.

Primary Actor: The primary actor is the WMS User.

Supporting Actors: The supporting actor is the Authentication System, as specified in [MS-AUTHSO].

Stakeholders and Interests: The WMS User, for deleting the CIM object.

Preconditions: The requested object cannot exist.

Minimal Guarantee: The failure of the deletion will be detected.

Success Guarantees: The WMS system guarantees that the failure of the deletion will be detected and the Windows Management Client will be notified.

Trigger: This use case is triggered by a request from the WMS User.

Main Success Scenario:

1. The **Windows Management Client** sends a request to the server specifying the name and class of the CIM object to be deleted.
2. The WMS detects the CIM object with the name and class supplied by the client. If the CIM object does not exist, then WMS returns an error.
3. The Server notifies the client about the failure of the deletion attempted on the CIM object.

Extensions: None.

4 System Context

This section describes the relationship between this system and its environment.

4.1 System Environment

The Windows Management System provides a way for WMS applications to access and manipulate CIM data on a remote server using a choice of three available network protocols. The system operates using a direct client-to-server communication channel.

The communication between the client and server components of the system can be carried out over a network connection that supports either DCOM over RPC (when using the WMI protocol) or SOAP over HTTP (when using the Web Services Management Protocol Extensions for Windows Server 2003 operating system (WSMAN) or Web Services Management Protocol Extensions for Windows Vista operating system (WSMV) protocols). Both protocols require that messages be able to be exchanged in a request/response fashion – that is, the network is configured in such a way that the client computer can send a message to the server and the server can send a message back to the client in response.

In order to offer access to CIM resources, a CIM Object Manager needs to be running on the server. The data managed by the CIMOM needs to be accessible to all of the system-defined protocols that are supported. For example, if a particular implementation of WMS supports both WMI and WSMV, then both protocols needs to be able to interact with the CIMOM in order to retrieve and manipulate data.

4.2 System Assumptions and Preconditions

WMS includes multiple protocols that provide access to CIMOM objects. An implementation may choose to support only a subset of the WMS protocols, but all supported protocols MUST operate on a common CIMOM store, so that changes using one protocol are visible to each of the others. If two protocols modify the same CIM instance nearly simultaneously, there is no guarantee that any particular one of the modifications will persist. There is also no guarantee that any one of the two protocol's updates to the instance (which may include changes to multiple object properties) will persist in full. The only guarantee is that, for each individual property of the CIM instance, exactly one of the attempted modifications will persist.

WMS assumes that the WMS Client already possesses certain data about the WMS Server:

- The client needs to know the IP address or host name of the server.
- The client needs to know the port number used to access CIMOM data. This endpoint may vary depending on the protocol being used.
- The client needs to know which of the three protocols are supported by the server, or implement a mechanism by which it can respond to the situation where a particular protocol is not supported by the server.
- The client needs to know the Uniform Resource Identifier (URI) used to access the desired managed object on the server.
- When using the WSMAN and WSMV protocols, if the client attempts to communicate over HTTPS it is assumed that the server is configured correctly to establish an HTTPS connection using a certificate.

- When using Kerberos for authentication, it is assumed that a key distribution center (KDC) is available and running, and that the client is configured to communicate with the KDC in order to authenticate requests.
- The client needs to know which CIM classes are supported on the server, or be able to respond to a missing class.
- The WSMAN and WSMV protocols do not provide any way for the client to retrieve information about any CIM object's class schema. This information is necessary in some circumstances – for example, in order to invoke a method on a CIM object, the client needs to know which methods (if any) are supported by the object's class and what parameters the method(s) require. If the client wishes to make use of the class schema of a CIM object, this information needs to already be known by the client.

4.3 System Relationships

The WMS System protocols are not technically dependent on each other. They are implemented independently of each other. Unless otherwise specified their only interdependencies are those of the individual protocols' underlying transport mechanisms, such as HTTP, RPC, or DCOM.

Because there are no logical relationships between the protocols in the WMS System, the following sections describe the protocol stacks for each of the protocols.

The Windows Management Instrumentation Remote Protocol communicates exclusively through DCOM. The DCOM Remote Protocol is the foundation for the Windows Management Instrumentation Remote Protocol and is used to establish the protocol, secure the communication channel, authenticate clients, and to implement a reliable communication between clients and servers.

The Web Services Management Protocol Extensions for Windows Server 2003 operating system Protocol and the Web Services Management Protocol Extensions for Windows Vista operating system use SOAP over HTTP or HTTPS for communication. The following figure depicts the protocols stack for the WMS System.

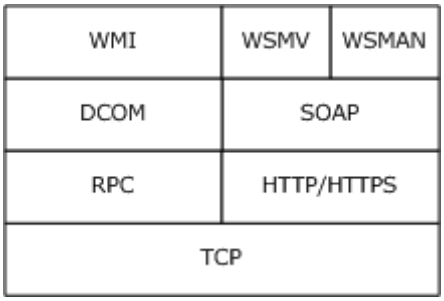

Figure 12: WMS System protocol stack

4.3.1 Black Box Relationship Diagram

At a high level, the WMS System provides a way for WMS applications to access and manipulate data. When viewed as an opaque entity, the system can be simply described as a mechanism through which applications access managed objects (represented as CIM data). The following diagram illustrates this concept.

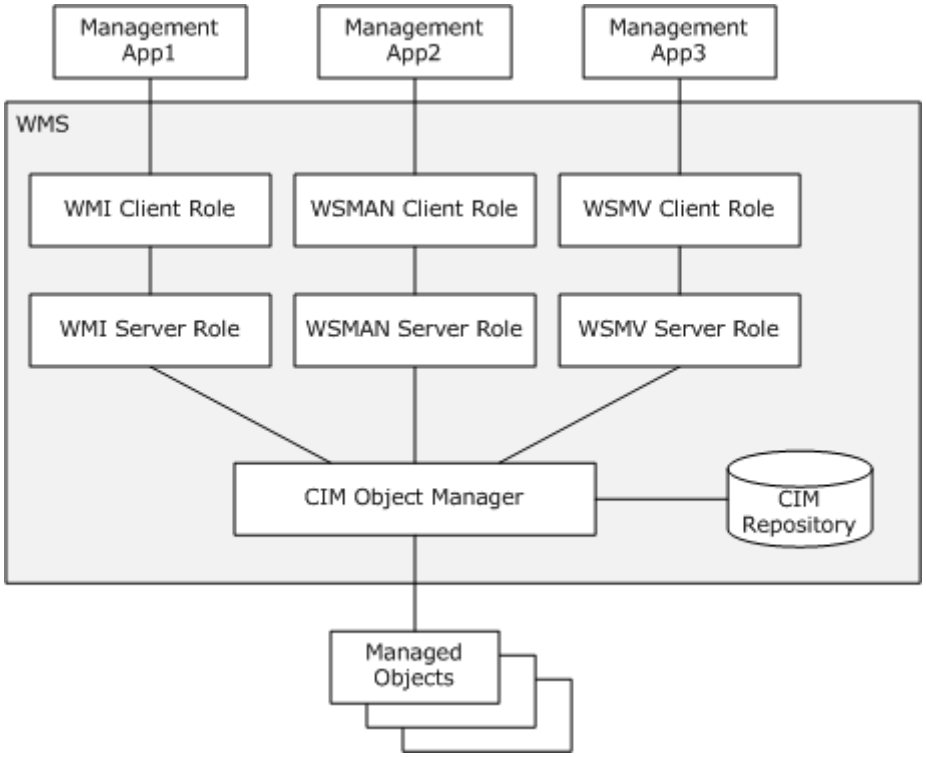

Figure 13: WMS System as CIM data interface

The applications in this diagram are WMS applications that are implemented to carry out one or more of the use cases described in Section 3.3.3 and 3.3.4. All of these tasks are accomplished through the retrieval and modification of data on the system that is to be managed. This data, which is represented using the CIM model, is made available to the applications through the WMS System.

On the remote end, this data is exposed to the WMS System as a set of managed objects. When an application interacts with the WMS System, it requests access to (or a specific modification of) these managed objects. The WMS System accepts this request, retrieves or modifies the appropriate data in the managed objects, and then returns the appropriate data to the application in response.

The specific details for the type and format of requests issued by the applications, and for the type and format of the responses issued back to the applications, are dependent on the individual protocol that is being used to carry out the action. Moreover, the actual capabilities of the system may differ based on the protocol being used. This detailed information is contained in the individual protocol specifications.

For details regarding the requests and responses issued when using the Windows Management Instrumentation Remote Protocol, see [MS-WMI].

For details regarding the requests and responses issued when using the Web Services Management Protocol Extensions for Windows Server 2003 operating system, see [MS-WSMAN].

For details regarding the requests and responses issued when using the Web Services Management Protocol Extensions for Windows Vista operating system, see [MS-WSMV].

4.3.2 System Dependencies

At a high level, there are two categories of dependencies upon which the WMS System relies:

- A network connection between the application and the managed objects.

If there is no network connection between the application and the managed objects to be retrieved or modified, the WMS System cannot function. This network connection needs to implement at least one of the three member protocols, described as follows:

- If a particular implementation of the WMS System implements the Windows Management Instrumentation Remote Protocol, then the network connection joining the applications and managed objects needs to implement all necessary underlying protocols (including RPC and DCOM).
- If a particular implementation of the WMS System implements the Web Services Management Protocol Extensions for Windows Server 2003 operating system or the Web Services Management Protocol Extensions for Windows Vista operating system, then the network connection joining the applications and managed objects needs to implement all necessary underlying protocols (including HTTP and SOAP).

For specific details regarding the necessary underlying protocols and network capabilities for the three member protocols of WMS, see the individual specifications.

The WMS System depends on the underlying security protocols to provide the support for authentication and authorization services. Refer to individual specifications for details on the specific security mechanisms used.

Furthermore, in order for the system to properly route traffic, the underlying network infrastructure needs to contain a DNS service that resolves a target host name to an IP address.

In order for the WMS System to retrieve and manipulate the managed objects, there needs to be a CIMOM running on the computer that facilitates this access. The CIMOM needs to be correctly configured to carry out the requests coming from the WMS System (on behalf of the applications on the other end).

4.3.3 System Influences

Though it is not necessary, an implementation of the WMS System MAY allow some protocol-specific configuration settings to be changed through Group Policy.<1>

4.4 System Applicability

The WMS system is primarily applicable in scenarios where centralized asset management, monitoring, and troubleshooting is desired.

4.5 System Versioning and Capability Negotiation

Although there are multiple possible combinations of underlying protocols supported by WMS, there is no system-wide capability negotiation mechanism. It is assumed that the WMS application knows which member protocols are supported by which computers.

The server **role** of all current versions of Windows supports only the mandated **ResourceURIs** and verbs. In particular, Windows Server operating system does not support the "Subscribe" verb.

The server protocols supported by specific Windows versions are listed in the following table.

Protocols implemented	Operating system versions
WMI	Windows 95 operating system, Microsoft Windows 98 operating system, Windows Millennium Edition operating system, Windows NT 4.0 operating system, Windows 2000 Server operating system, Windows XP operating system
WMI, WSMAN	Windows Server 2003 operating system
WMI, WSMV	Windows Vista operating system, Windows Server 2008 operating system, Windows 7 operating system, Windows Server 2008 R2 operating system

The client protocols supported by specific Windows versions are listed in the following table.

Protocols Implemented	Operating system versions
WMI	Windows 95, Windows 98, Windows Millennium Edition, Windows NT 4.0, Windows 2000 Server, Windows XP
WMI, WSMAN	Windows Server 2003
WMI, WSMV	Windows Vista, Windows Server 2008. Windows 7, Windows Server 2008 R2

4.6 System Vendor-Extensible Fields

There are two major extension points related to WMS: locale (for example, language) identifiers, and CIM schema. Neither of these concepts is included in WMS or defined by it, but WMS messages refer to them.

For WMS usage of locale identifiers, see [MS-WMI] section 3.1.4.1.4, [MS-WSMV] sections 3.1.4.1.8 and 3.1.4.1.9, and [MS-WSMAN] section 3.1.4.1.10.

A major purpose of WMS is to enable access to CIM objects, so references to CIM schema are pervasive in [MS-WMI], [MS-WSMV], and [MS-WSMAN], and in their normative references. For details of CIM schema, see [DMTF-DSP0004].

5 System Architecture

This section describes the basic structure of the system and the interrelationships among its parts, consumers, and dependencies.

5.1 Abstract Data Model

5.1.1 Server Abstract Data Model Diagram

The abstract data model for the WMS server role is illustrated in the following figure.

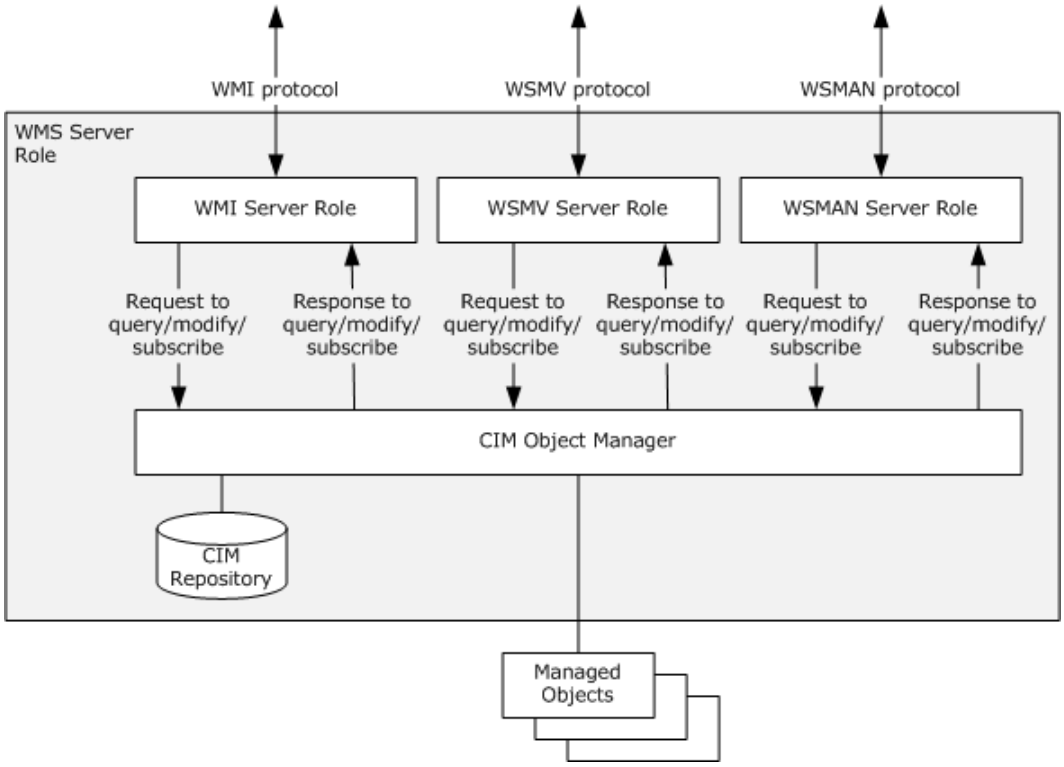

Figure 14: Server abstract data model

The WMS server MUST maintain at least one of the following abstract data models:

- WMI server ADM described in [MS-WMI] 3.1.1.
- RespondingSOAPNode role of the SOAP 1.2 HTTP binding defined in [SOAP1.2-2/2003] section 7, as constrained by [MS-WSMAN].
- RespondingingSOAPNode role of the SOAP 1.2 HTTP binding defined in [SOAP1.2-2/2003] section 7, as constrained by [MS-WSMV].

In addition, the WMS server has to maintain the following data:

- **NamespaceSDs**: a table of defined **CIM namespaces** and their associated **security descriptors** as defined in [MS-WMI] 5.2.

Whenever a CIM object is accessed by a component of the WMS System, the CIMOM needs to perform an access check against the security descriptor of the object's namespace. If the security descriptor denies the access, then the CIMOM does not access the CIM object, and instead returns an error to the requesting component. WMS uses **NamespaceSDs** to determine whether a particular component can access the CIMOM objects.

Creation or deletion of namespaces is implementation-specific. If the WMS server creates a namespace, it MUST create a corresponding row in the **NamespaceSDs** table. The access granted to a namespace is implementation-specific. Similarly, if a namespace is deleted, then its corresponding row in **NamespaceSDs** MUST be deleted as well.

Each CIM namespace on the server MUST define the CIM class __SystemSecurity defined in [MS-WMI] 2.2.30 for access to the namespace's security descriptor in NamespaceSDs.

See section 7.3 for more details.

5.1.2 Client Abstract Data Model

The WMS client MUST maintain at least one of the following abstract data models:

- WMI client ADM described in [MS-WMI] section 3.2.1.
- "RequestingSOAPNode" role of the SOAP 1.2 HTTP binding that is defined in [SOAP1.2-2/2003] section 7 and is constrained as specified in [MS-WSMAN].
- "RequestingSOAPNode" role of the SOAP 1.2 HTTP binding that is defined in section 7 of [SOAP1.2-2/2003] as constrained by [MS-WSMV].

5.2 White Box Relationships

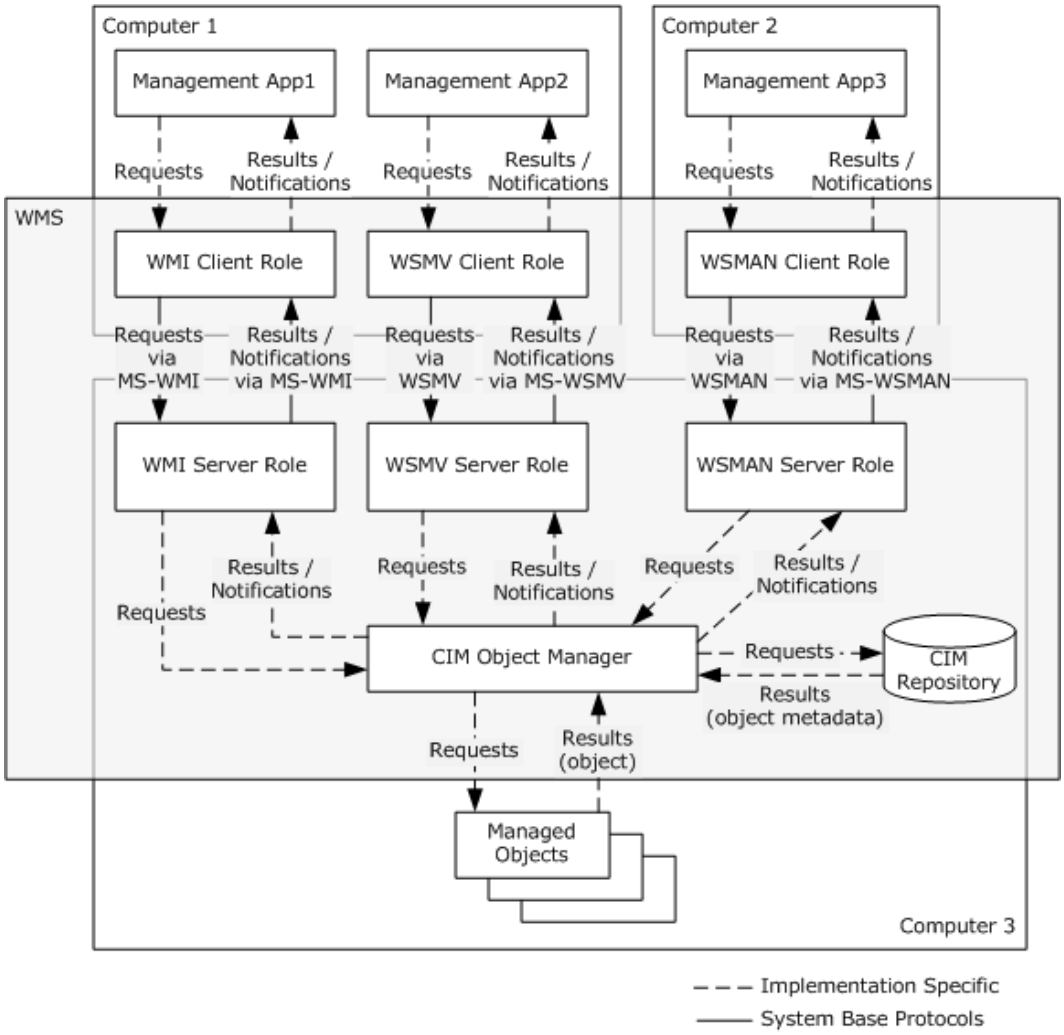

Figure 15: WMS White box relationships

The preceding figure represents the abstract model of the WMS System. One or more WMS applications use the system to access CIM objects on a set of managed computers. A single application may use one or more WMS protocols to access the server, depending upon the required capabilities.

5.3 Member Protocol Functional Relationships

5.3.1 Member Protocol Roles

The system defines only a single role for member protocols: a protocol for remote management, in all its forms, of CIM object data. The WSMAN, WSMV, and WMI protocols play this role.

The following table lists the WMS protocols and provides brief descriptions of each. Some major differences in capability between the protocols are highlighted.

External access protocols in WMS:

Protocol name	Protocol description	Document short name
Windows Management Instrumentation Remote Protocol	This is a DCOM-based protocol. Uses smaller messages than WSMV and WSMAN due to use of binary message encoding instead of SOAP. See [MS-WMI] section 2.2 for details of message encoding. Offers methods to query and modify the CIM classes and instances on a managed host.	[MS-WMI]
Web Services Management Protocol Extensions for Windows Server 2003 operating system	This protocol is Microsoft extensions to the Web Services for Management (WS-Management) Protocol. The protocol extends a prerelease draft version of WS-Management, and is incompatible with current DMTF specifications. This is an HTTP-based protocol which allows for easier network configuration than WMI when WMS applications and managed computers may be separated by a firewall. This protocol is based on a prerelease draft of the WS-Management specification, and is not compatible with the final 1.0 version. This protocol does not offer methods to query and modify the CIM classes on a managed host.	[MS-WSMAN]
Web Services Management Protocol Extensions for Windows Vista operating system	This protocol is Microsoft extensions to the Web Services for Management (WS-Management) Protocol. This protocol is based on version 1.0 of WS-Management. This is an HTTP-based protocol which allows for easier network configuration than WMI when WMS applications and managed computers may be separated by a firewall. The mandated level of support for event subscriptions is low. Qualifiers not defined in [DMTF-DSP0004] cannot be encoded in CIM Mapping. This protocol does not offer methods to query and modify the CIM classes on a managed host.	[MS-WSMV]

For full details of each protocol's capabilities, refer to their respective protocol specifications.

5.3.2 Member Protocol Groups

There are no functional group divisions among the member protocols.

5.4 System Internal Architecture

The flow of communication between the client and the server components of the system can be carried out over a network connection using any one of the member protocols (WMI, WSMAN and WSMV).

The conceptual framework for the Windows Management System is defined in section 4.3.1.

External entities to the WMS are the management applications which request CIM data from the WMS.

The Windows Management System is partitioned into the following components:

- CIM Object Manager: Serves as the interface to how CIM objects are accessed.
- CIM repository: Stores the metadata for the CIM classes.
- CIM managed objects: Hold the actual data.
- WMS server role.
- WMS client role.

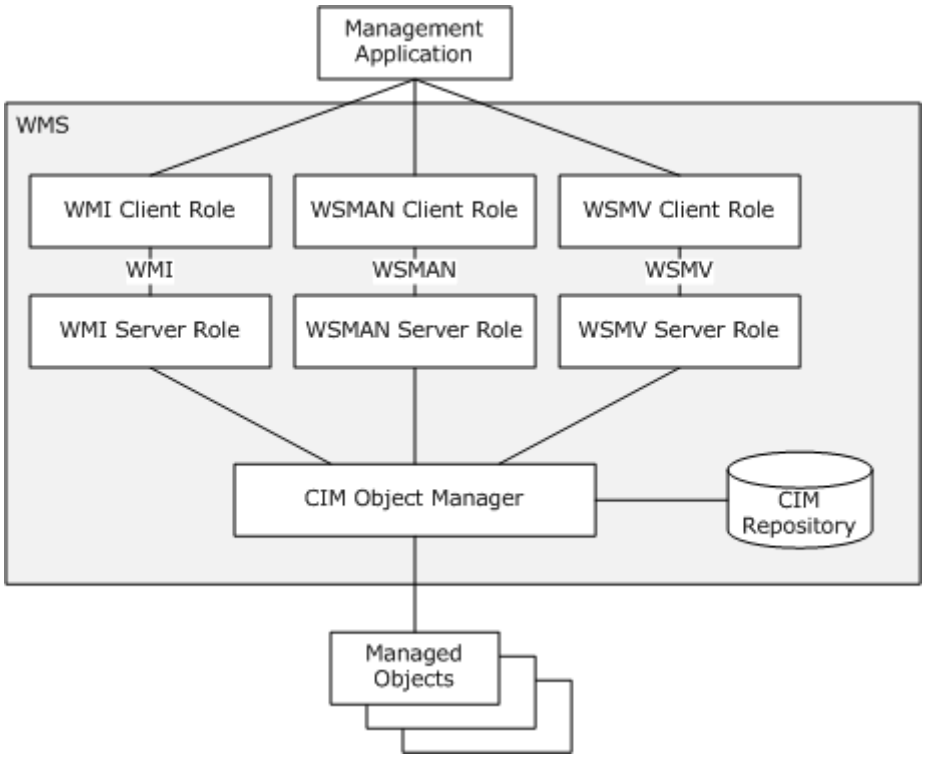

Figure 16: WMS internal architecture

5.5 Failure Scenarios

This section describes the common failure scenarios and the system behavior under those conditions.

5.5.1 Connection Breakdown Between the Entities

A common failure scenario is an unexpected connection breakdown between the client and server entities. A disconnection can be caused by the network not being available, or by one of the communicating participants becoming unavailable. In the case where the network is not available, both participants remain active and expect the other party to continue the communication pattern specified by the protocol being executed at the time of the failure. Similarly, in the case where one

of the participants is not available, the active participant expects the communication to proceed as specified by the protocol being executed.

Generally, a protocol detects a connection breakdown failure through either of the following methods:

- By using a timer object that generates an event if the corresponding participant has not responded within a reasonable time span.
- By being notified by the underlying protocol that the connection is disconnected.

When a connection disconnected event is detected, it causes the protocol to tear down all related communications and update any necessary data structures to maintain the system state.

Details about how connection breakdowns are handled in WMI can be found in [MS-WMI] sections 2.2.3, 2.2.8, and 2.2.11, and also in [MS-DCOM] and its dependent protocol specifications.

Details about how connection breakdowns are handled in WSMAN and WSMV can be found in [MS-WSMAN] sections 3.1.2 and 3.2.2, and also in [MS-WSMV].

5.5.2 Security Failures

Security failures are caused by not meeting the authentication or authorization requirements. In the case when the authentication or the authorization requirements are not met, the member protocol or the dependent protocols return an error to the participant. The participant will have to make appropriate changes to the security settings and retry the scenario as specified by the protocol.

Details about how security failures are handled can be found in [MS-WMI] (section 3.1.1.1.3), [MS-WSMV] (section 2.2.4.43), and also in [MS-WSMAN].

5.5.3 System Configuration Corruption and Other Internal Failures

A participant in the system could detect an unrecoverable internal state during its lifetime due to corruption of its configuration data, or due to the system being under high resource load. In such scenarios, if the system participant experiencing this problem decides that it cannot continue processing the specific type of future requests or any communication with any other entities, it can send an error to abort the call. If the operation originates from this participant, for example the server, then the operation should be attempted again after the appropriate changes, such as resetting the configuration data, are made to fix the internal error. If the operation originates from another participant, for example the client, that participant should retry the specific operation once the internal error is resolved.

Details about the how each protocol handles system errors can be found in [MS-WMI] section 2.2.11, [MS-WSMV] section 2.2.4.43, and [MS-WSMAN] section 2.2.4.1.

5.5.4 Other Common Failures in CIMOM Operations

In some cases, the server could return an error when performing an operation (such as executing a method on an object) due to the necessary class not being defined in a given namespace or due to the required information getting deleted due to other operations before this operation is executed.

In such cases, the server can send the appropriate errors to the client. The client can make calls after making necessary changes to the configuration on the server to get it to a state so the operation can succeed.

6 System Details

This section contains the details that complete the descriptions in earlier sections of the document. These details are needed to understand and implement this system.

6.1 Architectural Details

This section provides a series of examples illustrating the flow of communication between components of the WMS System, based on the type of operation being carried out. The examples are:

- A single request-response operation
- Enumeration of a large result set generated by a query
- Subscription models for subscribing to events

Each example is described in detail.

6.1.1 Single Request/Response Operations

A majority of the operations that are carried out within the WMS System are comprised of a single request and response. The following diagram illustrates the communication flow of these simple operations in the context of the system components described in section 5.2, using the Web Services Management Protocol Extensions for Windows Server 2003 Protocol. A detailed description of each message follows.

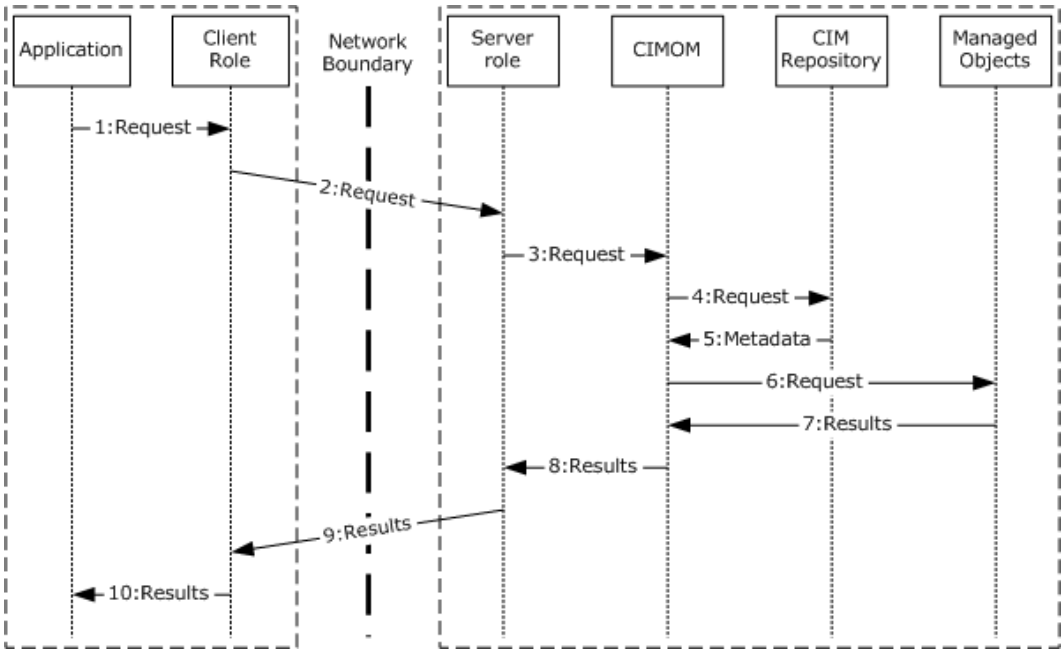

Figure 17: WMS communication flow

1:Request: To initiate the action, the client application issues a request through the component that implements the client role of the Web Services Management Protocol Extensions for Windows

Server 2003 Protocol. The request contains the necessary information to determine which operation is being carried out, but the exact data and information is implementation-specific.

2:Request: The client role of the member protocol sends the supplied information to the server role of the Web Services Management Protocol Extensions for Windows Server 2003 protocol. The request exact format and data contained in this request is dependent on the particular operation being carried out.

When the client application is retrieving data, the request contains the necessary routing information to direct the message to the correct endpoint and some identifier used to locate the specific data that is being requested (such as a particular instance of a particular CIM class). For examples of the exact message content and format, see [MS-WSMAN] section 4.1.1.

When the client application is modifying data, the request contains the necessary routing information to direct the message to the correct endpoint, some identifier used to locate the specific data that is being manipulated (such as a certain property of a particular instance of a particular CIM class), and the new values that the modified properties will take on. For examples of the exact message content and format, see [MS-WSMAN] section 4.1.3.

When the client application is invoking a method, the request contains the necessary routing information to direct the message to the correct endpoint, some identifier used to locate the specific method that is being invoked (such as a certain method of a particular CIM class), and any necessary parameters or variables that the method requires. For examples of the exact message content and format, see [MS-WSMAN] section 4.1.4.

3:Request: The server role of the MS-WSMAN protocol, upon receipt of the request from the client role, issues the request to the CIMOM. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols. The message contains essentially the same information as in **2:Request**, minus the routing information used to direct the message to the server role.

The CIMOM performs an access check to determine whether or not the requesting client is authorized to perform the action. The authorization depends on the operations being carried out: retrieval of data requires access to the object(s) as specified in [MS-WMI] section 3.1.4.3.4, modification of data requires access to the object(s) as specified in [MS-WMI] section 3.1.4.3.12, and invocation of methods requires access to the object(s) as specified in [MS-WMI] section 3.1.4.3.22.

4:Request: The CIMOM requests metadata from the CIM Repository, such as the schema of the requested object(s) and any information necessary to locate the particular object(s). The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

5:Metadata: The CIM Repository returns the requested metadata to the CIMOM. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

6:Request: The CIMOM sends a message requesting object(s) from the Managed Objects. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

7:Results: The Managed Objects return the requested object data to the CIMOM. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

8:Results: The CIMOM responds to the server role with the relevant information, based on the request that was issued.

9:Results: The server role of the MS-WSMAN protocol sends the supplied information to the client role of the MS-WSMAN protocol. The exact format and data that is sent depends on the type of operation being carried out.

When the client application requested to retrieve data, the response contains the necessary routing information to direct the message back to the client and the specific data that was requested (such as the property values of a particular instance of a particular CIM class). For examples of the exact message content and format, see [MS-WSMAN] section 4.1.1.

When the client application requested to modify data, the response contains the necessary routing information to direct the message back to the client, with either a simple notification of the success or failure of the operation or the actual data after the modification was applied (such as the changed property values of a particular instance of a particular CIM class). For examples of the exact message content and format, see [MS-WSMAN] section 4.1.3.

When the client application requested to invoke a method, the response contains the necessary routing information to direct the message back to the client, with either a simple notification of the success or failure of the operation or some particular return data that is relevant to the method that was invoked (such as the return value of a particular method of a particular CIM class). For examples of the exact message content and format, see [MS-WSMAN] section 4.1.4.

10:Results: The client role of the Web Services Management Protocol Extensions for Windows Server 2003 protocol, upon receipt of the response from the server role, delivers the response to the application. The message contains essentially the same information as in **9:Results**, minus the routing information used to direct the message to the server role.

6.1.2 Enumerations

When enumerating a set of managed objects, the result set may be too large to fit in a single response message. Hence, enumerations consist of more than a single request/response pair. The following diagram illustrates the communication flow of an enumeration operation, in the context of the system components described in section 5.2. A detailed description of each message follows.

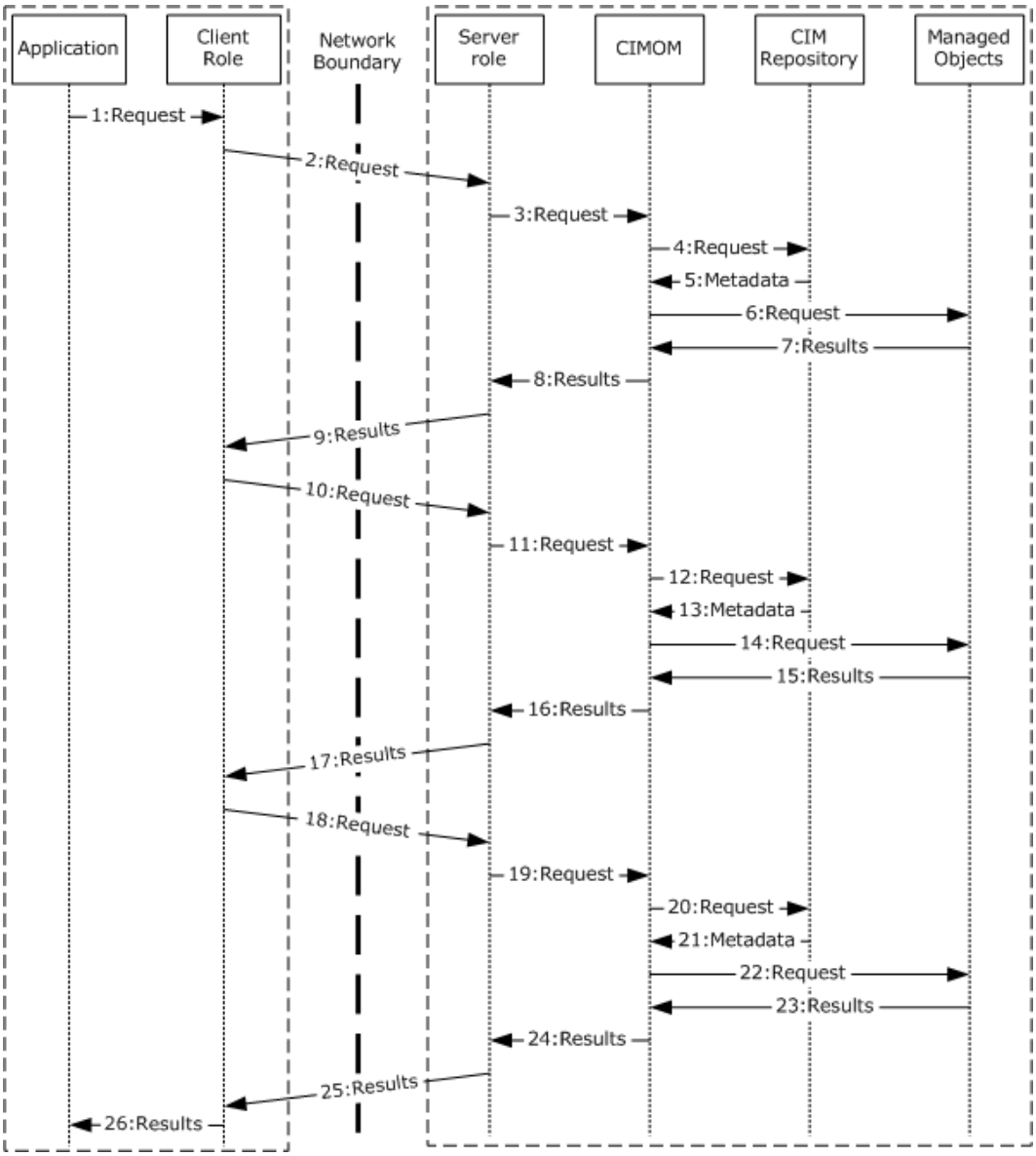

Figure 18: WMS communication flow in enumeration operation

1:Request: To initiate the enumeration, the client application issues a request through the component that implements the client role of the particular member protocol being used. The request contains the following information:

- Routing information that is necessary to direct the message to the correct endpoint;
- An identifier that is used to locate the specific dataset that is being requested, such as a specific CIM class; and
- An optional filter that is used to select specific result objects from within the specified data set; for example, a **WMI Query Language (WQL)** query that chooses individual CIM objects based on a particular property value.

2:Request: The client role of the member protocol sends the supplied information to the server role of the member protocol. This message contains the same information as **1:Request**.

For examples of the exact message content and format, see [MS-WSMV] section 4.1.2.1.

3:Request: The server role of the member protocol, upon receipt of the request from the client role, issues the request to the CIMOM. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols. The message contains essentially the same information as in **2:Request**, minus the routing information used to direct the message to the server role.

The CIMOM performs an access check to determine whether or not the requesting client is authorized to perform the action. The enumeration of data requires access to the object(s) as specified in [MS-WMI] section 3.1.4.3.16.

4:Request: The CIMOM requests metadata from the CIM Repository, such as the schema of the requested object(s) and any information necessary to locate the particular object(s). The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

5:Metadata: The CIM Repository returns the requested metadata to the CIMOM. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

6:Request: The CIMOM sends a message requesting the desired object(s) from the Managed Objects. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

7:Results: The Managed Objects return the requested object data to the CIMOM. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

8:Results: The CIMOM responds to the server role with the relevant information, based on the request that was issued. The response contains the necessary routing information to direct the message back to the client, with context information used to identify the particular enumeration in subsequent requests.

9:Results: The server role of the member protocol sends the supplied information to the client role of the member protocol. This message contains the same information as **8:Results**. For details on the exact message content and format, see [MS-WSMV] section 4.1.2.2.

10:Request: The client role of the member protocol issues a request to retrieve the objects being enumerated. The request contains the necessary routing information to direct the message to the correct endpoint, along with the contextual information received in **9:Results** that is used to identify which data to be returned at the server side. For details on the exact message content and format, see [MS-WSMV] section 4.1.2.3.

11:Request: The server role of the member protocol uses the context information to determine which objects need to be retrieved from the CIMOM. The server role then issues the request to the CIMOM. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols. The message contains whatever information is necessary to retrieve the relevant data objects according to the enumeration request. It is important to note that this set of objects may not be the entire set of objects that will be returned through the enumeration - in this example case, they are not.

12:Request: The CIMOM requests metadata from the CIM Repository, such as the schema of the requested object(s) and any information necessary to locate the particular object(s). The exact

format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

13:Metadata: The CIM Repository returns the requested metadata to the CIMOM. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

14:Request: The CIMOM sends a message requesting the desired object(s) from the Managed Objects. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

15:Results: The Managed Objects return the requested object data to the CIMOM. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

16:Results: The CIMOM responds to the server role with the requested set of data objects. The response contains the necessary routing information to direct the message back to the client, along with context information used to identify the next set of objects to be returned to this enumeration in subsequent requests. For details on the exact message content and format, see [MS-WSMV] section 4.1.2.4.

17:Results: The server role of the member protocol sends the supplied information to the client role of the member protocol. This message contains the same information as **16:Results**.

18:Request: The client role of the member protocol issues a request to retrieve more of the objects being enumerated. The request contains the necessary routing information to direct the message to the correct endpoint, along with the contextual information received in **17:Results** that is used to identify which data to be returned at the server side. For details on the exact message content and format, see [MS-WSMV] section 4.1.2.5.

19:Request: The server role of the member protocol uses the context information to determine which objects need to be retrieved from the CIMOM. The server role then issues the request to the CIMOM. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols. The message contains whatever information is necessary to retrieve the relevant data objects according to the enumeration request. In this example case, these returned objects are the last to be returned to the particular enumeration.

20:Request: The CIMOM requests metadata from the CIM Repository, such as the schema of the requested object(s) and any information necessary to locate the particular object(s). The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

21:Metadata: The CIM Repository returns the requested metadata to the CIMOM. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

22:Request: The CIMOM sends a message requesting the desired object(s) from the Managed Objects. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

23:Results: The Managed Objects return the requested object data to the CIMOM. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

24:Results: The CIMOM responds to the server role with the requested set of data objects. The response contains the necessary routing information to direct the message back to the client, along

with a notification that there are no more CIM objects that will be returned for this particular enumeration. For details on the exact message content and format, see [MS-WSMV] section 4.1.2.6.

25:Results: The server role of the member protocol sends the supplied information to the client role of the member protocol. This message contains the same information as **24:Results**.

26:Results: The client role of the member protocol, upon receipt of the response from the server role, delivers the response to the application because the response indicated that all of the enumerated data has been returned. The message contains all of the enumerated CIM objects. It is important to clarify that this message is not sent over a network connection, so it is not split into separate messages like those messages exchanged between the client and server roles that carried the CIM object data. Logically, the data is sent as one cohesive piece – the method in which this data is actually transmitted from the client role of the member protocol to the application is implementation-dependent.

6.1.3 Pull Event Subscriptions

The following diagram illustrates the communication flow of subscription and event delivery with pull subscriptions, in the context of the system components described in section 5.2. A detailed description of each message follows. The communication flow of subscription and event delivery is based on invoking synchronous operations described in [MS-WMI] section 3.2.4.2.8.

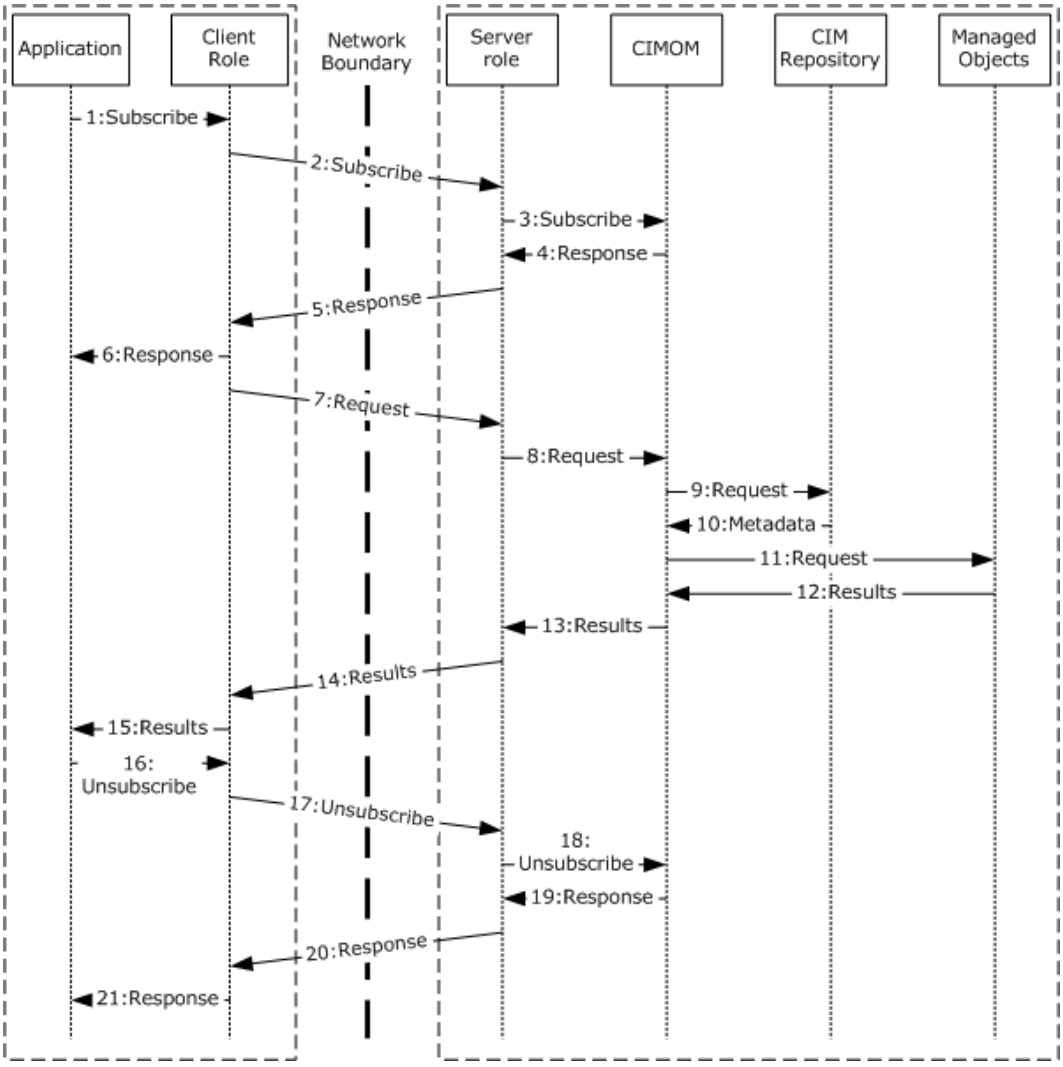

Figure 19: Communication flow of subscription and event delivery with pull subscriptions

1:Subscribe: To initiate the subscription, the client application issues a request through the component that implements the client role of the particular member protocol being used. The request contains the necessary routing information to direct the message to the correct endpoint, along with the necessary information to define the specific set of events that should be delivered (such as a particular CIM class and a filter that selects only critical error events).

2:Subscribe: The client role of the member protocol sends the supplied information to the server role of the member protocol. This message contains the same information as **1:Subscribe**.

For details, see [MS-WMI] section 3.1.4.3.20.

3:Subscribe: The server role of the member protocol sends the supplied information to the CIMOM, in order to register the subscription. This message contains the same information as **2:Subscribe**.

4:Response: The CIMOM registers the subscription, and then sends a response to the server role's subscribe request, indicating the success or failure of the subscribe request and any additional necessary data, such as bookmark information used when retrieving future events.

5:Response: The server role sends a response to the client role's subscribe request, indicating the success or failure of the subscribe request. This message contains the same information as **4:Response**.

For details on the exact message content and format, see [MS-WMI] section 3.1.4.3.20.

6:Response: The client role sends a message to the application, indicating the success or failure of the subscription.

7:Request: The client role of the member protocol sends a message to the server role of the member protocol, containing the necessary routing information to direct the message to the correct endpoint (such as what was included in **1:Subscribe**) and any additional necessary data that was included in **5:Response**.

For details, see [MS-WMI] section 3.2.4.2.8.

8:Request: The server role of the member protocol being used issues a query to the CIMOM in order to retrieve events that have occurred. The contents and format of this message are implementation-dependent. For example, the server role could simply retrieve all events from the CIMOM and filter them after receipt, or the CIMOM could expose an interface such that the server role could request particular events based on the subscription information.

9:Request: The CIMOM requests event metadata from the CIM Repository. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

10:Metadata: The CIM Repository returns the requested metadata to the CIMOM. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

11:Request: The CIMOM sends a message requesting the desired events from the Managed Objects. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

12:Results: The Managed Objects return the requested event data to the CIMOM. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

13:Results: The CIMOM returns the requested events to the server role of the member protocol. The exact format of this message is implementation-dependent.

14:Results: The server role of the member protocol sends the events to the client role of the member protocol. The response contains the necessary routing information to direct the message to the client role, the actual events being delivered, and optionally some bookmark information used to determine which events should be returned in response to the next request for events (within the same subscription).

For details on the exact message content and format, see [MS-WMI] section 3.1.4.3.20.

15:Results: The client role of the member protocol delivers the actual events to the application. The format of this message is implementation-dependent.

16:Unsubscribe: When it no longer wishes to receive events, the application issues a request to the client role of the WMI protocol to unsubscribe. The exact data and format of this request are implementation-specific.

17:Unsubscribe: The client role of the WMI protocol sends the supplied information to the server role of the WMI protocol. The request contains the necessary routing information to direct the message to the server role, along with whatever information is necessary to identify the specific subscription that is to be canceled.

For details on the exact message content and format, see [MS-WMI] section 3.1.4.4.

18:Unsubscribe: The server role of the WMI protocol sends the supplied information to the CIMOM. This message contains the same information as **17:Unsubscribe**.

19:Response: The CIMOM clears any stored subscription information and deletes the subscription. It then sends a response back to the server role, indicating the success or failure of the subscription cancellation.

20:Response: The server role of the member role sends a response back to the client role, indicating the success or failure of the subscription cancellation.

For details on the exact message content and format, see [MS-WMI] section 3.1.4.3.20.

21:Response: The client role reports the success or failure of the subscription cancellation to the application. The format of this message is implementation-dependent.

6.1.4 Push Event Subscriptions

The following diagram illustrates the communication flow of subscription and event delivery with push subscriptions, in the context of the system components described in section 5.2, using the WMI protocol. The communication flow of subscription and event delivery is based on asynchronous operations described in [MS-WMI] section 3.2.4.2.9. A detailed description of each message follows.

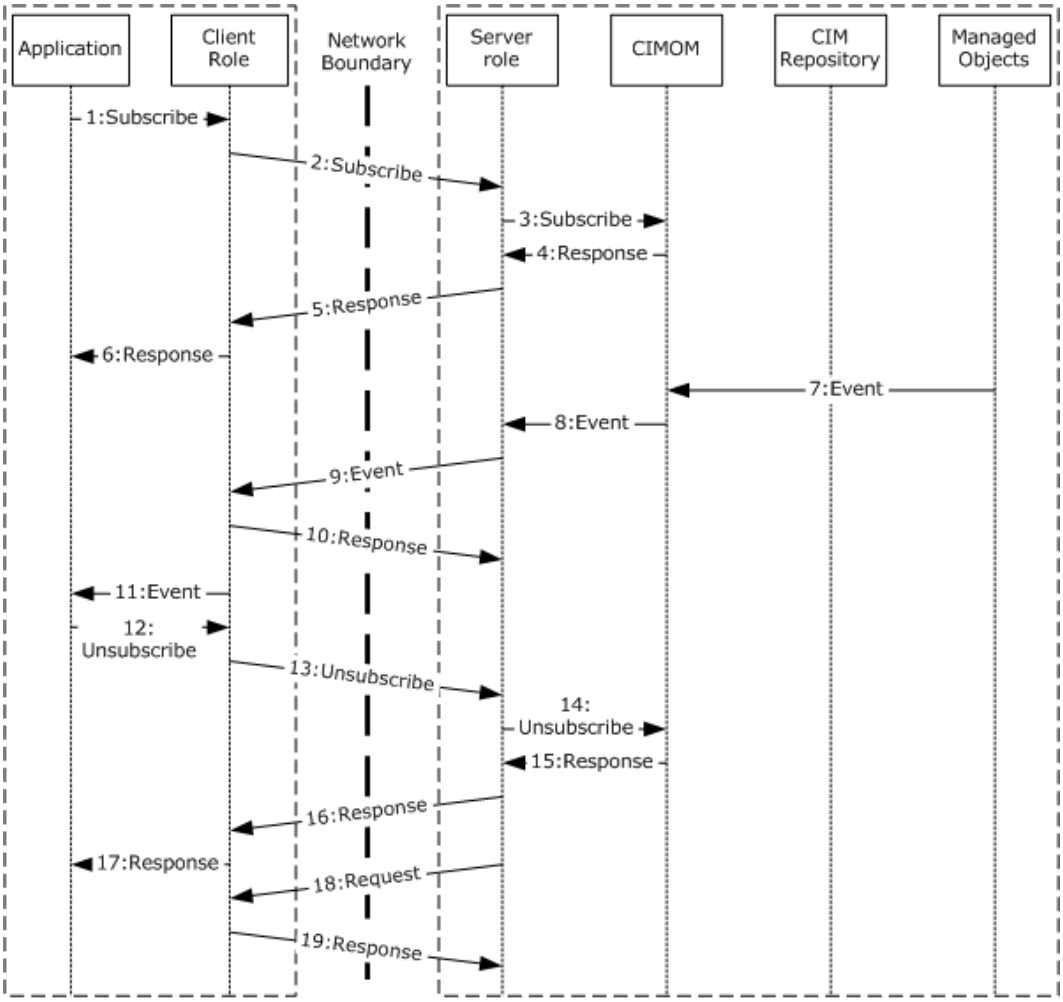

Figure 20: Communication flow of subscription and event delivery with push subscriptions

1:Subscribe: To initiate the subscription, the client application issues a request through the component that implements the client role of the WMI protocol. The exact data and format of this request are implementation-specific.

2:Subscribe: The client role of the WMI protocol sends the supplied information to the server role of the WMI protocol. The request contains the necessary routing information to direct the message to the correct endpoint, along with the necessary information to define the specific set of events that should be delivered, such as a particular CIM class and a filter that selects only critical error events.

For details on the exact message content and format, see [MS-WMI] section 3.1.4.3.21.

3:Subscribe: The server role of the member protocol sends the supplied information to the CIMOM, in order to register the subscription. This message contains the same information as **2:Subscribe**.

4:Response: The CIMOM registers the subscription, and then sends a response to the server role's subscribe request, indicating the success or failure of the subscribe request and any additional necessary data, such as information needed to identify the specific subscription that is to be canceled when sending an Unsubscribe request.

5:Response: The server role sends a response to the client role's subscribe request, indicating the success or failure of the subscribe request. This message contains the same information as **4:Response**.

For details on the exact message content and format, see [MS-WMI] section 3.1.4.3.21.

6:Response: The client role sends a message to the application, indicating the success or failure of the subscription.

7:Event: When an event is generated, the Managed Objects sends the event to the CIMOM. The content and format of this message are implementation-dependent.

8:Event: The CIMOM determines whether or not the event in **7:Event** needs to be delivered based on the registered subscriptions. If so, the event handed off to the server role of the member protocol. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

9:Event: The server role of the WMI protocol sends the event to the client role. This message contains the call to the **IWbemObjectSink::Indicate()** method, defined in [MS-WMI] section 3.1.4.2.1, along with the actual event being delivered.

10:Response: The client role of the WMI protocol sends a message back to the server role of the WMI protocol in order to send the response to **IWbemObjectSink::Indicate()** defined in [MS-WMI] section 3.1.4.2.1.

11:Event: The client role of the member protocol delivers the actual events to the application. The format of this message is implementation-dependent.

12:Unsubscribe: When it no longer wishes to receive events, the application issues a request to the client role of the member protocol to unsubscribe. The request contains the necessary routing information to direct the message to the server role, along with whatever information is necessary to identify the specific subscription that is to be canceled.

For details on the exact message content and format, see [MS-WMI] section 3.1.4.3.2.

13:Unsubscribe: The client role of the member protocol sends the supplied information to the server role of the member protocol. This message contains the same information as **12:Unsubscribe**.

14:Unsubscribe: The server role of the member protocol sends the supplied information to the CIMOM. This message contains the same information as **18:Unsubscribe**.

15:Response: The CIMOM clears any stored subscription information and deletes the subscription. It then sends a response back to the server role, indicating the success or failure of the subscription cancellation.

16:Response: The server role of the member role sends a response back to the client role, indicating the success or failure of the subscription cancellation.

For details on the exact message content and format, see [MS-WMI] section 3.1.4.3.2.

17:Response: The client role reports the success or failure of the subscription cancellation to the application. The format of this message is implementation-dependent.

18:Request: Once the subscription is canceled, a call is made from the server role to the client described in [MS-WMI] section 3.1.4.2.2.

19:Response: The client role sends back the response to the call made in **18:Request**.

6.1.5 Publisher-Initiated Event Subscriptions

It is important to note that the WMI and WSMAN protocols do not support publisher-initiated event subscriptions; the following section is only relevant when using the WSMV member protocol.

The following diagram illustrates the communication flow of subscription and event delivery with publisher-initiated subscriptions, in the context of the system components described in section 5.2. A detailed description of each message follows.

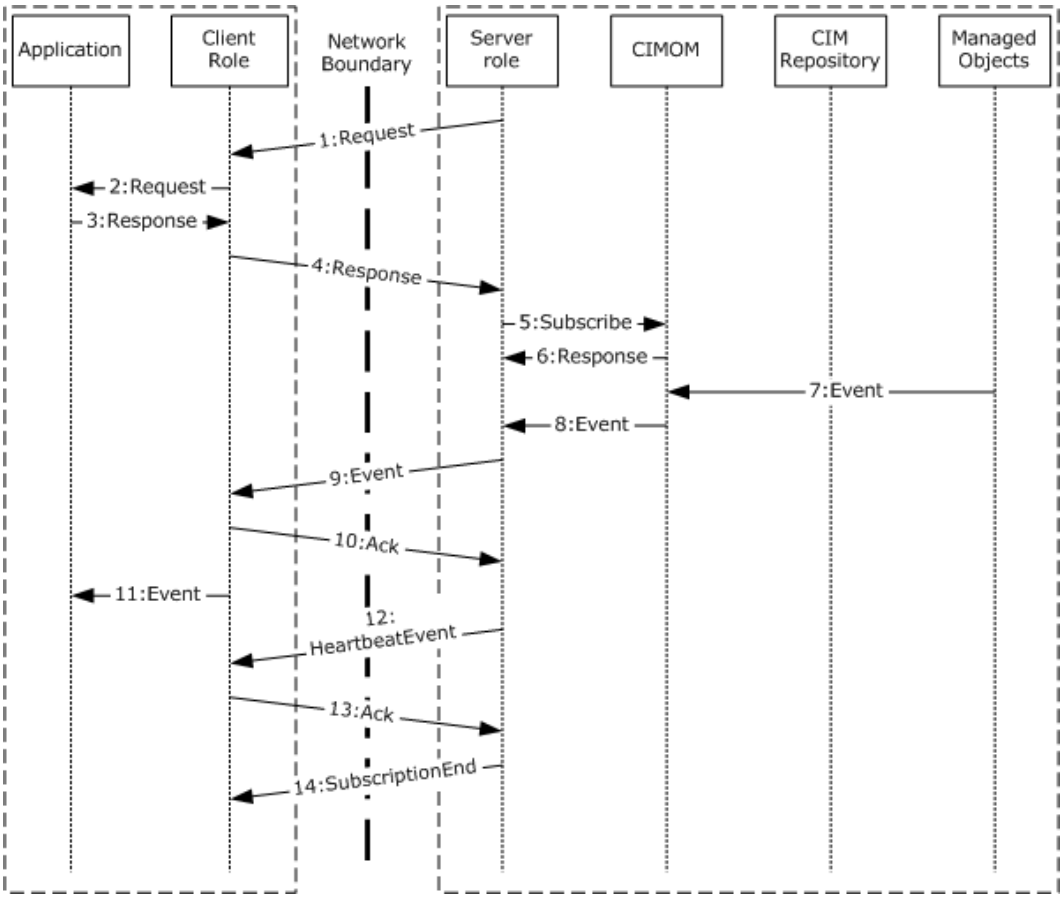

Figure 21: Communication flow of subscription and event delivery with publisher-initiated subscriptions

1:Request: In this scenario, the event publisher initiates the subscription. The server role of the member protocol issues an enumeration request to the client role of the member protocol. The identity and routing information of the client role is determined based on implementation-specific local configuration at the server role. This request contains the necessary routing information to direct the message to the client role, along with optional filters to scope the actual subscriptions that are returned.

For details on the exact message content and format, see [MS-WSMV] section 3.1.4.1.30.

2:Request: The client role of the member protocol issues a request to the application in order to retrieve the active subscriptions. This data is generated in an implementation-specific fashion, and the format and content of this message is implementation-dependent.

3:Response: The application returns information about the active subscriptions to the client role of the member protocol. The request contains, for each active subscription, the information needed to be able to deliver events, such as CIM class names or filters, along with the necessary routing information to direct the message to the server role.

For details on the exact message format, see [MS-WSMV] section 3.1.4.1.30.

4:Response: The client role of the member protocol sends the supplied information to the server role of the member protocol. This message contains the same information as **3:Response**.

5:Subscribe: The server role of the member protocol sends the supplied information to the CIMOM, in order to register the subscription. This message contains the same information as **4:Response**.

6:Response: The CIMOM registers the subscription, and then sends a response to the server role's subscribe request, indicating the success or failure of the subscribe request and any additional necessary data, such as bookmark information used when retrieving future events.

7:Event: When an event is generated, the Managed Objects sends the event to the CIMOM. The content and format of this message are implementation-dependent.

8:Event: The CIMOM determines whether or not the event in **7:Event** needs to be delivered based on the stored subscription information. If so, the event handed off to the server role of the member protocol. The exact format of this message is dictated by the CIMOM implementation, and is not specified by any of the individual member protocols.

9:Event: The server role of the member protocol sends the event to the client role. This message contains the necessary routing information to direct the message to the client role, along with the actual event being delivered.

For details on the exact message content and format, see [MS-WSMV] section 3.1.4.1.30.

10:Ack: The client role of the member protocol optionally sends a message back to the server role of the member protocol in order to acknowledge the successful receipt of the delivered events. This message contains whatever information is necessary to allow the server role to determine which event delivery it is in response to.

For details on the exact message content and format, see [MS-WSMV] section 3.1.4.1.30.

11:Event: The client role of the member protocol delivers the actual events to the application. The format of this message is implementation-dependent.

12:HeartbeatEvent: In order to ensure that the connection between the client role and the server role does not terminate due to inactivity, if an event has not occurred for a long period of time the server role of the WSMV protocol sends a heartbeat message to the client role of the WSMV protocol. This message follows the same format as a typical Event notification; the "hearbeat" itself is considered an event.

For details on the exact message content and format, see [MS-WSMV] section 3.1.4.1.30.

13:ACK: The client role of the member protocol optionally sends a message back to the server role of the member protocol in order to acknowledge the successful receipt of the delivered heartbeat event. This message contains whatever information is necessary to allow the server role to determine which event delivery it is in response to.

For details on the exact message content and format, see [MS-WSMV] section 3.1.4.1.30.

14:SubscriptionEnd: When the publisher terminates the subscription, the server role of the WSMV protocol issues a SubscriptionEnd message to the client role of the member protocol. The message contains the necessary routing information to direct the message to the client role, along with whatever information is necessary to identify the specific subscription that is to be canceled.

For details on the exact message content and format, see [MS-WSMV] section 3.1.4.1.30.

6.1.6 Protocol-dependent Differences in Eventing behavior

WMI offers the capability to notify a subscriber for any event it is interested in. Client applications can subscribe to receive events by providing a query that describes the events and establishing a subscription. WMI uses the WMI Query Language (WQL) to submit event queries and defines the type of events to be returned. For details, see [MS-WMI] sections 3.1.4.3.20 and 3.1.4.3.21.

If the DCOM connection between client and server is broken, then the server will cancel the query. Once the client detects the broken connection, it can register the query again, but will not receive events generated between the time the first query is canceled and the second query is registered.

WSMV and its underlying protocols define several ways for a client to be notified of events; for details, see [MS-WSMV] section 3.1.4.1.30. However, only a few ResourceURIs are mandatory, and the set of mandated verbs for each ResourceURI is limited; see [MS-WSMV] section 3.1.4.1 for details.

WSMAN and its underlying protocols define several ways for a client to be notified of events; for details, see [MS-WSMAN] section 3.1. However, only a few ResourceURIs are mandatory, and the set of mandated verbs for each ResourceURI is limited; see [MS-WSMAN] section 3.1.4.1 for details.

6.2 Communication Details

The system does not define additional communication details beyond those described in the specifications of the protocols supported by the system.

6.3 Transport Requirements

The system does not define a transport beyond those described in the specifications of the protocols supported by the system, as listed in section 5.1.

6.4 Timers

There are no timer events beyond those specified in the member protocol documents.

6.5 Non-Timer Events

The system does not define any non-timer events beyond those specified in the underlying member protocol documents.

6.6 Initialization and Reinitialization Procedures

The system does not define any initialization or reinitialization procedures beyond those specified in the underlying member protocol documents.

6.7 Status and Error Returns

The system does not define any status and error returns beyond those specified in the underlying member protocol documents.

7 Security

This section documents system-wide security issues that are not otherwise described in the Technical Documents (TDs) for the Member Protocols. It does not duplicate what is already in the Member Protocol TDs unless there is some unique aspect that applies to the system as a whole.

The WMS System provides access to a common set of data through three different protocols, which access the data over a remote network connection. Therefore, there are three main aspects of security in the system: the security configuration for each individual protocol, the security of the data as it is sent over a network, and the security of the CIM data itself.

The following diagram illustrates the logical points at which the various security configurations are applied.

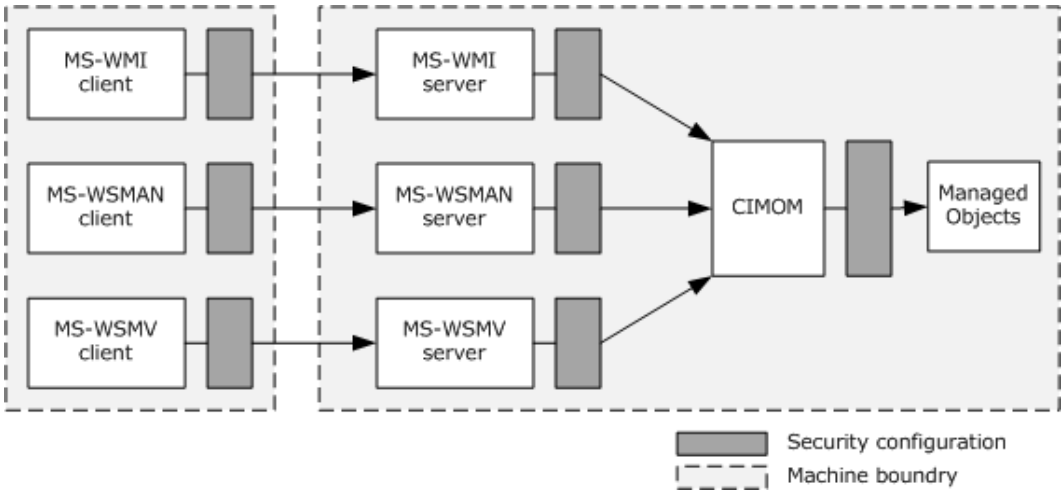

Figure 22: Security configuration settings

7.1 Security Configuration Per Protocol

There are a number of security settings defined in the individual member TDs that only secure access through the individual member protocol. It is important to note that the security configuration settings in this section only apply to the relevant individual protocol, and they do not secure access to CIM resources through either of the other two protocols.

The WSMAN protocol MUST authenticate all requests using one of a few possible configured security profiles, as specified in [MS-WSMAN] section 5.1. This ensures that the identity of a user issuing a request is known.

The WSMAN protocol MUST authorize all requests based on a specified **Security Descriptor Description Language (SDDL)** string, as specified in [MS-WSMAN] section 5.1. This ensures that users are only able to access CIM resources if they have been granted permission for remote access through the WSMAN protocol.

The WSMV protocol MUST authenticate all requests using one of a few possible configured security profiles, as specified in [MS-WSMV] section 5.1. This ensures that the identity of a user issuing a request is known.

The WSMV protocol MUST authorize all requests based on a specified SDDL string, as specified in [MS-WSMV] section 5.1. This ensures that users are only able to access CIM resources if they have been granted permission for remote access through the WSMV protocol.

The WMI protocol assumes, as a prerequisite, that clients using the protocol possess valid **credential**s that are recognized by the server, and that they use security providers that recognize such credentials to authenticate the user, as specified in [MS-WMI] section 1.5.

7.2 Security of Data Over the Network

Because the CIM data being transferred across a network connection can potentially hold sensitive information, it is important to secure it from tampering or accidental disclosure. This is accomplished by encrypting the data, leaving it unreadable to malicious third parties.

The WSMAN protocol supports the transport of messages using HTTPS, as documented in [MS-WSMAN] section 1.4. Traffic sent over HTTPS is encrypted, and it is only able to be decrypted by the proper receiving party.

The WSMV protocol supports the transport of messages using HTTPS, as documented in [MS-WSMV] section 1.4. Traffic sent over HTTPS is encrypted, and it is only able to be decrypted by the proper receiving party.

The WSMV protocol also supports the transport of encrypted messages using HTTP as a transport, as documented in [MS-WSMV] section 2.2.9.1. This allows messages to be encrypted in situations when HTTPS encryption is not possible, such as when the required certificates are not deployed.

The WMI protocol supports the transport of messages using the DCOM Remote Protocol as documented in [MS-WMI] section 2.1.

DCOM specifies a set of constants that convey the level of authentication.

7.3 Security of CIM Data

Regardless of the particular protocol used for remote access, the CIMOM restricts access to the underlying managed objects. This is accomplished by assigning SDDL strings to a particular CIM namespace, requiring that any user accessing data in a namespace have the appropriate access rights. For example, the security descriptor can enable a particular user to read data from a namespace, but not to modify it – in that case, the user would be allowed to retrieve a managed object but not to set a value on that managed object.

The various access rights that can be defined for a particular user or group are specified in [MS-WMI] section 5.2. To query or change the security descriptor associated with a particular namespace, the **GetSD** and **SetSD** methods of the **__SystemSecurity** class MUST be used, as specified in [MS-WMI] sections 3.1.4.3.22 and 3.1.4.3.23.

In order to query or change the security descriptor associated with a particular namespace using the WSMAN protocol, the **GetSD** and **SetSD** methods of the **__SystemSecurity** class can be invoked by setting the **Action URI** to appropriately reference the **__SystemSecurity** class and desired method, as specified in [MS-WSMAN] section 3.1.4.

In order to query or change the security descriptor associated with a particular namespace using the WSMV protocol, the **GetSD** and **SetSD** methods of the **__SystemSecurity** class can be invoked by setting the Action URI to appropriately reference the **__SystemSecurity** class and desired method, as specified in [MS-WSMV] section 3.1.4.

Additionally, a CIMOM can employ any arbitrary additional, implementation-specific security restrictions and access checks. Any added security measures are not defined by this system, and are purely up to the CIMOM implementation. Modifications made by one member protocol to a namespace's security descriptors are visible to other member protocols.

8 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Windows 95 operating system
- Microsoft Windows 98 operating system
- Windows Millennium Edition operating system
- Windows NT 4.0 operating system
- Windows 2000 Server operating system
- Windows XP operating system
- Windows Server 2003 operating system
- Windows Vista operating system
- Windows Server 2008 operating system
- Windows 7 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 4.3.3: Windows exposes the following configuration settings through Group Policy that affect access to CIM resources through WSMAN or WSMV:

- Allow/disallow the various supported authentication mechanisms
- Allow/disallow unencrypted traffic
- Specify a list of server hostnames that a client can connect to
- Enable/disable the automatic listening to network traffic (server side)
- Specify the channel-binding token (CBT) hardening level

These settings are specified under the Windows component Windows Remote Management (WinRM).

9 Appendix B: Enumerating Class Schema

The following script shows how to enumerate the current class schemas on a Windows machine:

```
EnumerateSchema "root"

sub EnumerateSchema(ns)
     if instr(ns,"LDAP") = 0 then
                              wscript.echo "#pragma namespace(""\\\\.\\" & escapeit(ns) &
""")"
     set wmi = getobject("winmgmts:\\.\" & ns)
     for each cls in wmi.subclassesof("")
             wscript.echo cls.getobjecttext_(0)
     next

     for each subns in wmi.instancesof("__namespace")
             EnumerateSchema ns & "\" & subns.name
     next
     end if
end sub

function escapeit(ns)
     escapeit = replace(ns, "\", "\\")
end function
```

To use the script:

1. Save the script as "getmofs.vbs" on the target machine.
2. From a cmd.exe window, type "cscript getmofs.vbs > schema.mof". On Windows versions that support the Windows Integrity Mechanism, including the Windows Vista operating system and later versions, the CMD window must be "elevated", that is, run with administrative privileges.

The resulting output (in schema.mof) represents all of the class schemas on the particular system.

10 Appendix C: Enumerating Namespace Access Control Lists

The following script can be used to enumerate the security **access control list (ACL)** for all namespaces on a Windows system:

```
if wscript.arguments.count = 0 then
wscript.echo "Usage: cscript getnsacls.vbs <computer>"
wscript.echo "Script needs to be run on a Vista+ machine to resolve the ACL to SDDL"
wscript.quit 1
end if

computer = wscript.arguments(0)
set helper = getobject("winmgmts:win32_securitydescriptorhelper")

getacls("root")

sub getacls(ns)
myNS = "\\" & computer & "\" & ns
wscript.echo myNS

set wmi = getobject("winmgmts:{authenticationlevel=pktPrivacy}!" & myNS)
set security = wmi.get("__systemsecurity=@")
retval = security.getSD(SDarray)
retval = helper.binarySDtoSDDL(SDarray,SDDL)
wscript.echo SDDL

for each subns in wmi.instancesof("__namespace")
getacls ns & "\" & subns.name
next
end sub
```

This script depends on functionality available in Windows Vista operating system, Windows Server 2008 operating system, Windows Server 2008 R2 operating system and Windows 7 operating system to format the security descriptor into a readable format. The script can be used by doing the following, provided that the current user credentials of the person running the script are those of an administrator on the remote computer:

- Save the script as "getnsacls.vbs".
- Execute the script against a remote computer by typing **cscript getnsacls.vbs <Computer>** at a cmd.exe prompt where <*Computer*> is the name of the remote computer.

11 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

12 Index