[MS-WMI]:
Windows Management Instrumentation Remote Protocol Specification

This protocol documentation is intended for use in conjunction with publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

A protocol specification does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them.

Revision Summary

Date	Revision History	Revision Class	Comments
07/20/2007	0.1	Major	MCPP Milestone 5 Initial Availability
09/28/2007	0.2	Minor	Updated the technical content.
10/23/2007	0.3	Minor	Updated the technical content.
11/30/2007	1.0	Major	Updated and revised the technical content.

Date	Revision History	Revision Class	Comments
01/25/2008	1.1	Minor	Updated the technical content.

Table of Contents

1 Introduction

Windows Management Instrumentation (WMI) is a Distributed Component Object Model, as specified in [MS-DCOM], a client/server–based framework that provides an open and automated means of systems management. WMI leverages the **Common Information Model (CIM)**, as specified in [DMTF-DSP004], to represent various components of the operating system. CIM is the conceptual model for storing enterprise management information. The information available from CIM is specified by a series of classes and associations, and the elements contained therein (methods, properties, and references). These constructs describe the data available to WMI clients.

1.1 Glossary

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

Activation
Authentication Level
Common Information Model (CIM)
Common Information Model (CIM) Class
Common Information Model (CIM) Instance
Common Information Model (CIM) Method
Common Information Model (CIM) Namespace
Common Information Model (CIM) Object
Common Information Model (CIM) Path
Common Information Model (CIM) Property
Common Information Model (CIM) Relative Path
CLSID
Extrinsic Event
Interface Definition Language (IDL)
Interface Pointer
Intrinsic Event
Manageable Entity
Microsoft Interface Definition Language (MIDL)
Opnum
Release
Remote Procedure Call (RPC)
Security Principal
Security Provider
Superclasses and Subclasses
Universally Unique Identifier (UUID) or Globally Unique Identifier (GUID)

The following terms are specific to this document:

CIM Localizable Information: The portion of information in a CIM class definition that could be language- or country-specific.

Client: In the context of this document, "client" is used to identify the system consuming WMI services and initiating [MS-DCOM] calls into WMI servers.

Empty CIM Object: A data structure confirming to the WMI serialization model as having no properties, no method, and no derivation.

Server: In the context of this document, "server" is used to identify the system implementing WMI services, providing management services and accepting [MS-DCOM] calls from WMI clients.

WMI Asynchronous Operation: An operation executed on the server side. The client continues executing and does not check whether there is a response available from the server.

WMI Semisynchronous Operation: An operation executed on the server side while the client is regularly checking whether there is a response available from the server.

WMI Synchronous Operation: An operation executed on the server side while the client is waiting for the response message.

WQL (WMI Query Language): A subset of the American National Standards Institute Structured Query Language (ANSI SQL). WQL differs from the standard SQL in that it retrieves from classes rather than tables, and returns CIM classes or instances rather than rows. WQL is specified in section 2.2.1.1.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

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.

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997, http://www.opengroup.org/public/pubs/catalog/c706.htm

[DMTF-DSP004] 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

[FIPS127] Federal Information Processing Standards Publication, "Database Language SQL", FIPS PUB 127, June 1993, http://www.itl.nist.gov/fipspubs/fip127-2.htm

[IEEE754] Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", IEEE 754-1985, October 1985, http://grouper.ieee.org/groups/754/

[MS-DCOM] Microsoft Corporation, "Distributed Component Object Model (DCOM) Remote Protocol Specification", March 2007.

[MS-DTYP] Microsoft Corporation, "Windows Data Types", January 2007.

[MS-ERREF] Microsoft Corporation, "Windows Error Codes", January 2007.

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MS-GPOL] Microsoft Corporation, "Group Policy: Core Protocol Specification", June 2007.

[MS-LCID] Microsoft Corporation, "Windows Language Code Identifier (LCID) Reference", July 2007.

[MS-OAUT] Microsoft Corporation, "OLE Automation Protocol Specification", March 2007.

[MS-RPCE] Microsoft Corporation, "Remote Procedure Call Protocol Extensions", January 2007.

[MS-WMIO] Microsoft Corporation, "Windows Management Instrumentation Encoding Version 1.0 Protocol Specification", September 2007.

[RFC1001] Network Working Group, "Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods", RFC 1001, March 1987, http://www.ietf.org/rfc/rfc1001.txt

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

[RFC4234] Crocker, D., Ed. and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005, http://www.ietf.org/rfc/rfc4234.txt

[UNICODE] The Unicode Consortium, "Unicode Home Page", 2006, http://www.unicode.org/

1.2.2 Informative References

[DFS-R WMI] Microsoft Corporation, "DFSR WMI Classes", http://msdn2.microsoft.com/en-us/library/bb540028.aspx

Note This WMI management interface is exposed as a public API in Windows Server 2003 R2, Windows Vista and Windows Server 2008 operating system. A .mof file publishing the capabilities of the WMI provider associated with DFS-R is included in the DFS-R distribution under %%SYSTEM_ROOT%%\webm\dfsrprov.mof.

[MSDN-WQL] Microsoft Corporation, "Querying with WQL", http://msdn2.microsoft.com/en-us/library/aa392902.aspx

[MSDN-WMISYSCLASSES] Microsoft Corporation, "WMI System Classes", http://msdn2.microsoft.com/en-us/library/aa394583.aspx

1.3 Protocol Overview (Synopsis)

The Windows Management Instrumentation Remote Protocol is the Microsoft implementation of the Common Information Model (CIM), as specified in [DMTF-DSP004]. The Windows Management Instrumentation Remote Protocol uses CIM as the conceptual model for representing enterprise management information that can be managed by a Windows administrator.

The Windows Management Instrumentation Remote Protocol is implemented in Windows in a three-tier architecture, as illustrated in the following figure.

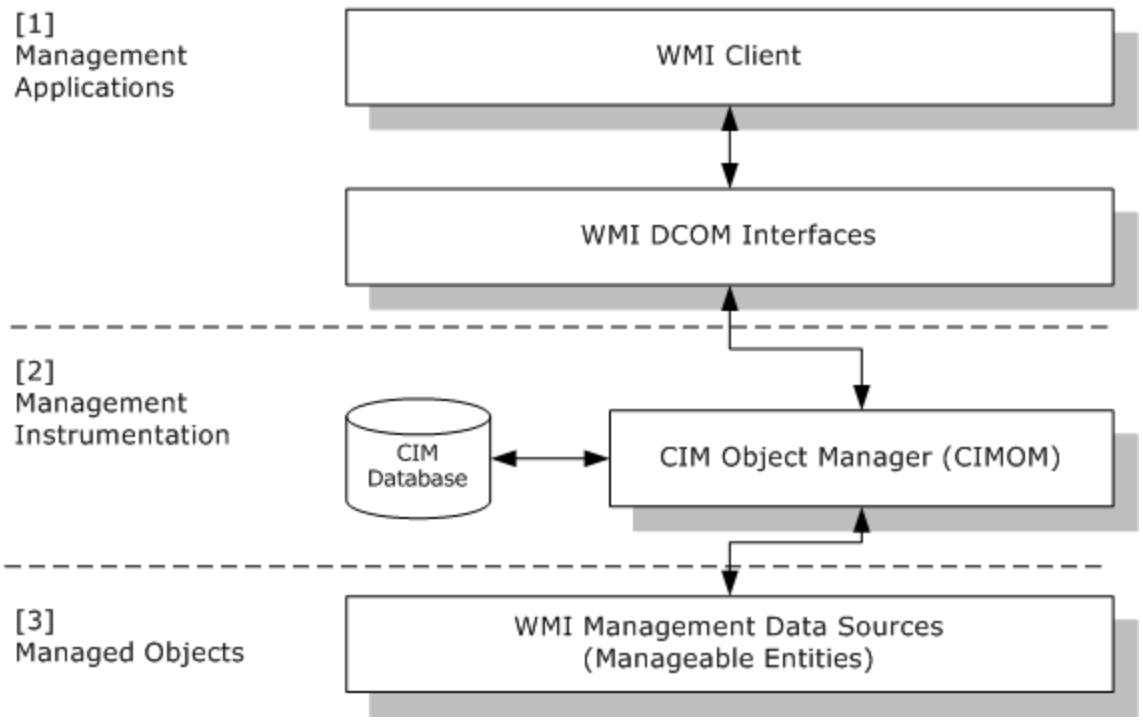


Figure 1: Windows Management Instrumentation Remote Protocol architecture

At layer 3, the Windows Management Instrumentation Remote Protocol management data sources are designed to interact locally with the Windows **manageable entities**. Layer 2 supports the core of the Windows Management Instrumentation Remote Protocol service and is called the CIM Object Manager (CIMOM). CIMOM interacts with the Windows Management Instrumentation Remote Protocol management data sources and the database storing and caching **CIM class** definitions and **CIM instances** from the [DMTF-DSP004]. Layer 1 implements the Distributed Component Object model (as specified in [MS-DCOM]) interfaces used by the Windows Management Instrumentation Remote Protocol to communicate over the network between Windows Management Instrumentation Remote Protocol clients and servers. This layer is the only layer communicating over the network. The network communication MUST be achieved by using the Distributed Component Object Model (DCOM) Remote Protocol and a set of Windows Management Instrumentation Remote Protocol DCOM interfaces, as specified in [MS-WMI].

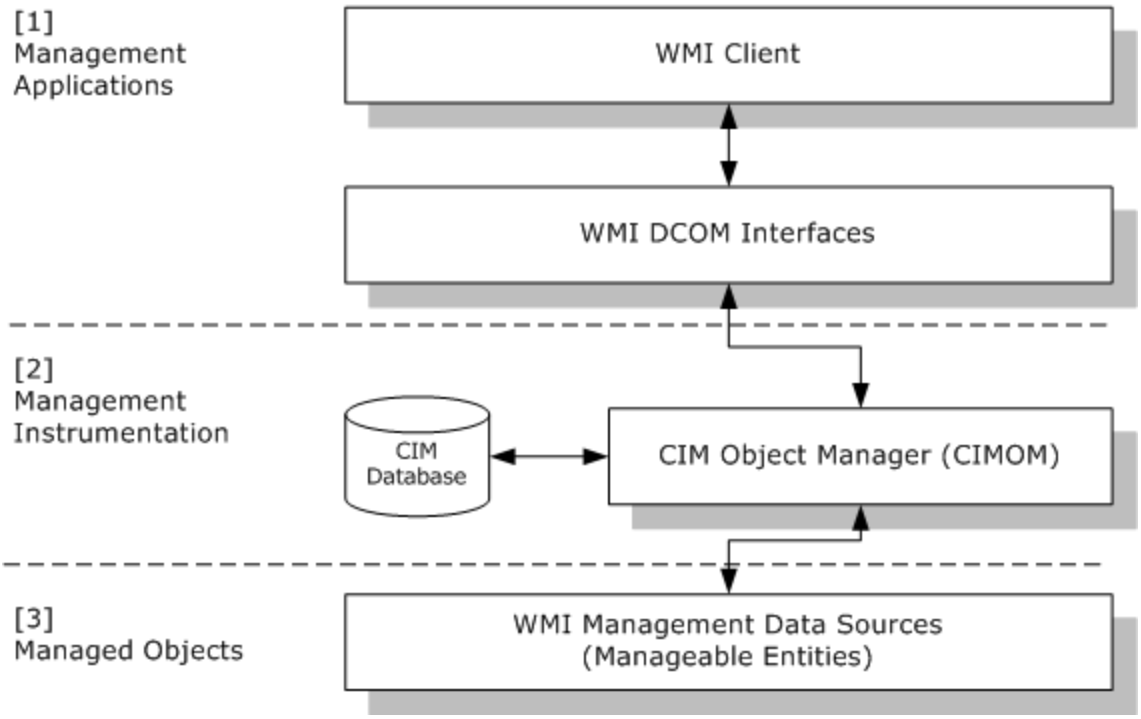


Figure 2: Clients can be local or remote from the server

Windows Management Instrumentation Remote Protocol clients can be local or remote from the server, as illustrated in the preceding figure. In either case, the same set of Windows Management Instrumentation Remote Protocol interfaces is used. The server makes no differentiation between a local client and a remote client. Likewise, if the client is running on the server, the client makes no differentiation either. The communication works the same way between clients and server; all interactions between clients and server MUST always be made through the DCOM Remote Protocol locally or remotely. Therefore, clients are always acting in a message submission mode through the DCOM Remote Protocol to leverage the Windows Management Instrumentation Remote Protocol interfaces implemented on the server side.

The management information exchanged between clients and server (and server and clients) is transmitted over the network by the Windows Management Instrumentation Remote Protocol as a custom-marshaled payload, as specified in [MS-DCOM].

The Windows Management Instrumentation Remote Protocol serializes the management information transmitted, as specified in [MS-WMIO]. The understanding of this document requires a working knowledge of the concepts, structures, and communication protocols as specified in [MS-DCOM], [DMTF-DSP004], and [MS-WMIO] before entering into the reading of this Windows Management Instrumentation Remote Protocol document. Namespace security MUST be controlled by using security descriptors, as specified in [MS-DTYP].

1.4 Relationship to Other Protocols

The Windows Management Instrumentation Remote Protocol MUST use the DCOM Remote Protocol to communicate over the network and authenticate all requests issued against the infrastructure. The DCOM Remote Protocol is actually the foundation for the Windows Management Instrumentation Remote Protocol and is used to accomplish the following:

- Establish the protocol.
- Secure the commutation channel.
- Authenticate clients.
- Implement a reliable communication between clients and servers.

This implies that the DCOM Remote Protocol implementation MUST provide and MUST use all underlying protocols, as specified in [MS-RPCE], [MS-DCOM], and [C706].

Besides DCOM Remote Protocol support, the Windows Management Instrumentation Remote Protocol uses a special encoding, as specified in [MS-WMIO], to transfer information as specified in [DMTF-DSP004] over the network.

Other protocols that are built on top of the Windows Management Instrumentation Remote Protocol include:

- Microsoft WS-Management extensions, as specified in [MS-WSMAN].
- Microsoft Corporation, "Windows Group Policy Protocols" as specified in [MS-GPOL].

1.5 Prerequisites/Preconditions

The client using the protocol MUST be in possession of valid credentials recognized by the server accepting the client requests. The client MUST use **security providers** that recognize such credentials to authenticate to the remote server by using SSPI supported by the Remote Procedure Call Protocol.

The server system MUST be started with the DCOM Remote Protocol **activation** service fully initialized before the activation request. The client MUST be configured to receive activation requests from the server if it wants to call the service **asynchronously**, as specified in section 4.4.

An implementation of the DCOM Remote Protocol MUST be available.

1.6 Applicability Statement

The Windows Management Instrumentation Remote Protocol implementation is designed for managing Windows components represented by CIM classes on remote clients and servers. The Windows Management Instrumentation Remote Protocol is not intended to act as a transport protocol for large amounts of data.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas. The Windows Management Instrumentation Remote Protocol does explicit negotiation, as follows:

- The client of this protocol MUST use the mechanism, as specified in [MS-DCOM] section 1.7, to discover what interfaces are supported by the exported object and to interpret E_NOINTERFACE result, as specified in [MS-DCOM] section 1.7. Based on the availability of the requested interface, the client should adjust, as specified in sections 3.1.5.1, 3.1.5.2, and 3.1.5.3.
- The protocol uses return codes as a capability discovery mechanism, and the client SHOULD interpret them as a capability negotiation, as specified in section 3.1.5.5.1. Windows behavior is documented in the respective sections.

1.8 Vendor-Extensible Fields

Vendors MAY use the Windows Management Instrumentation Remote Protocol to extend the CIM schema with Windows by using operations as specified in section 3.1.5.2.

This protocol uses HRESULT values as specified in [MS-ERREF]. Vendors can define their own HRESULT values, provided they set the C bit (0x20000000) for each vendor-defined value, indicating the value is a customer code.

1.9 Standards Assignments

There are no standards assignments for this protocol. This protocol uses the following **CLSIDs** (as specified in [MS-DCOM] section 1.9):

- CLSID_WbemLevel1Login ({8BC3F05E-D86B-11D0-A075-00C04FB68820})
- CLSID_WbemBackupRestore ({C49E32C6-BC8B-11D2-85D4-00105A1F8304})

The following **GUIDs** are used for the interfaces:

- IID_IWbemLevel1Login ({F309AD18-D86A-11d0-A075-00C04FB68820})
- IID_IWbemLoginClientID ({d4781cd6-e5d3-44df-ad94-930efe48a887})
- IID_IWbemLoginHelper ({541679AB-2E5F-11d3-B34E-00104BCC4B4A})
- IID_IWbemServices ({9556DC99-828C-11CF-A37E-00AA003240C7})
- IID_IWbemBackupRestore ({C49E32C7-BC8B-11d2-85D4-00105A1F8304})
- IID_IWbemBackupRestoreEx ({A359DEC5-E813-4834-8A2A-BA7F1D777D76})
- IID_IWbemClassObject ({DC12A681-737F-11CF-884D-00AA004B2E24})
- IID_IWbemContext uses ({44aca674-e8fc-11d0-a07c-00c04fb68820})

2 Messages

The following sections specify how Windows Management Instrumentation Remote Protocol messages are transported and Windows Management Instrumentation Remote Protocol message syntax.

2.1 Transport

Windows Management Instrumentation Remote Protocol messages MUST be transported via the DCOM Remote Protocol. The Windows Management Instrumentation Remote Protocol MUST use the dynamic endpoints allocated and managed by the DCOM Remote Protocol infrastructure.

The client connection MAY be secured at different **authentication levels** negotiated by the DCOM Remote Protocol infrastructure.

2.2 Message Syntax

2.2.1 Common Data Types

2.2.1.1 WQL Query

A client has the capability to express a query against a server. This query MUST be expressed in **WQL**. WQL is a subset of the American National Standards Institute Structured Query Language, as specified in [FIPS127] and [MSDN-WQL]. WQL differs from the standard SQL in that it retrieves from classes rather than tables, and returns CIM classes or CIM instances rather than rows. It supports a specific semantic designed to query against CIM classes or CIM instances with their related characteristics. Queries MAY be of the following three forms:

- Schema queries: Queries focused on CIM classes.
- Data queries: Queries focused on CIM instances.
- Event queries: Queries focused on events triggered by state changes of CIM classes or CIM instances. Events triggered on CIM instances can be internal to the infrastructure (intrinsic) or external to the infrastructure (extrinsic). Events can also be timer events.

WQL uses terminologies and concepts, as specified in [DMTF-DSP004], and requires familiarity with the CIM model.

The next section specifies the complete syntax of WQL queries for schema, data, and event queries.

2.2.1.1.1 WQL Schema and Data Query

The syntax for the WQL schema and data queries is provided in Augmented BNF (ABNF) format.

```
; ------------------------------------
; WQL schema and data queries
; ------------------------------------

DATA-WQL =
 ("SELECT" <PROPERTY-LIST> "FROM" <CLASS-NAME>
   <OPTIONAL-SEL-WHERE>)/
 ("SELECT" ASTERISK "FROM" <CLASS-NAME> <OPTIONAL-SEL-WHERE>)/
 ("SELECT" ASTERISK "FROM META_CLASS" <OPTIONAL-META-WHERE>)/
```

```
 ("ASSOCIATORS OF {" <OBJECT-REL-PATH> "}"
   <OPTIONAL-ASSOC-WHERE>)/
 ("REFERENCES OF {" <OBJECT-REL-PATH> "}" <OPTIONAL-REF-WHERE>)

PROPERTY-LIST = <PROPERTY-NAME> <PROPERTY-LIST2>
PROPERTY-LIST2 = [COMMA <PROPERTY-LIST>]

OPTIONAL-SEL-WHERE = ["WHERE" <EXPR>]
OPTIONAL-META-WHERE = ["WHERE __THIS ISA" <CLASS-NAME>]
OPTIONAL-ASSOC-WHERE =
 ["WHERE" [ "AssocClass=" <CLASS-NAME> BLANK ]
 [ "RequiredAssocQualifier=" <QUALIFIER-NAME> BLANK ]
 [ "RequiredQualifier=" <QUALIFIER-NAME> BLANK ]
 [ "ResultClass=" <CLASS-NAME> BLANK ]
 [ "ResultRole=" <PROPERTY-NAME> BLANK ]
 [ "Role=" <PROPERTY-NAME> BLANK ]
 [ "KeysOnly" BLANK ]
 [ "ClassDefsOnly" BLANK ]
                       ]
OPTIONAL-REF-WHERE =
 ["WHERE" [ "RequiredQualifier=" <QUALIFIER-NAME> BLANK ]
 [ "ResultClass=" <CLASS-NAME> BLANK ]
 [ "Role=" <PROPERTY-NAME> BLANK ]
 [ "KeysOnly" BLANK ]
 [ "ClassDefsOnly" BLANK ]
                     ]

OBJECT-REL-PATH =
 <CLASS-NAME> "=" <TYPED-CONSTANT> <OBJECT-REL-PATH2>
OBJECT-REL-PATH2 =
 [COMMA <OBJECT-REL-PATH>]

; ------------------------------------
; Expression
; ------------------------------------

EXPR =
 ( [OPEN-PARENTHESIS] <PROPERTY-EVALUATION>
    <EXPR2> [CLOSE-PARENTHESIS] ) /
     ( [OPEN-PARENTHESIS] "__CLASS" <EQUIVALENT-OPERATOR>
       <CLASS-NAME> <EXPR2> [CLOSE-PARENTHESIS] )

EXPR2 = ( ["OR"  [OPEN-PARENTHESIS] <EXPR> [CLOSE-PARENTHESIS] ])/
        ( ["AND" [OPEN-PARENTHESIS] <EXPR> [CLOSE-PARENTHESIS] ])

PROPERTY-EVALUATION =
   ( <PROPERTY-NAME> <OPERATOR> <TYPED-CONSTANT> ) /
    ( <PROPERTY-NAME> <IS-OPERATOR> "NULL" )

OPERATOR = <EQUIVALENT-OPERATOR> /
           <COMPARE-OPERATOR>

EQUIVALENT-OPERATOR = "=" / "!="
COMPARE-OPERATOR = "<=" / ">=" / "<" / ">" / "LIKE"
IS-OPERATOR = "IS" / "IS NOT"

; ------------------------------------
; Characters
```

```
; ----------------------------------

ALPHA = %x41-5A
DIGIT = %x30-39
COMMA = ","
ASTERISK = "*"
OPEN-PARENTHESIS = "("
CLOSE-PARENTHESIS = ")"
BLANK = " " / "\x09"

STRING-IDENTIFIER = ALPHA *(ALPHA / DIGIT / (*("_") ALPHA / DIGIT))

CLASS-NAME = [__]<STRING-IDENTIFIER>
PROPERTY-NAME = [__]<STRING-IDENTIFIER>
QUALIFIER-NAME = <STRING-IDENTIFIER>


TYPED-CONSTANT = INT /
                 REAL /
                 STRING /
                 DATETIME /
                 BOOL

INT = "[-+]?\d+"
REAL = "[-+]?(\d*\.\d+)|(\d+)"
STRING = ["]([a-z][A-Z]\d)*["]
DATETIME =
 "(\d\d\d\d)(0\d|1[012])(0\d|[12][0-9]|3[01])([0-1]\d|2[0-3])
  ([0-5]\d)([0-5]\d)[.]\d\d\d\d\d\d[+-]([0-6][02468][0]|7[0-2][0])"

BOOL = "TRUE" / "FALSE"
```

Schema objects and keywords	Description
CLASS-NAME	Identifies the CIM class name to be queried.
PROPERTY-NAME	Identifies the name of a property of the CIM class.
QUALIFIER-NAME	In the context of a WQL query, QUALIFIER-NAME is an attribute of a PROPERTY-NAME defining the nature of an association with another CIM class.
DATA-WQL	A string expressing the WQL query. The WQL string uses different WQL reserved keywords to select the type of information desired.
SELECT	A keyword expressing the selection of information requested (similar to SQL SELECT). SELECT expresses the CIM class or CIM instance to be queried. It MUST be specified when the ASSOCIATORS OF or the REFERENCES OF keyword is not used. It MUST NOT be used when the ASSOCIATORS OF or the REFERENCES OF keyword is used.
PROPERTY-LIST	A list of PROPERTY-NAME values. PROPERTY-NAME values in the list MUST be separated by a comma (",").
ASTERISK	Requires all properties of a CIM class or a CIM instance.

Schema objects and keywords	Description
FROM	A keyword that MUST be specified with the SELECT statement to express the CIM class or CIM instance the query MUST be executed against.
OPTIONAL-SEL-WHERE	The WHERE statement narrows the scope of a SELECT.
OPTIONAL-META-WHERE	The WHERE statement narrows the scope of a SELECT. The WHERE statement followed by the __THIS ISA statement is narrowing the scope of the WQL query to return CIM instances only made out of the CLASS-NAME specified.
__CLASS	A keyword referring to the **CIM object**, indicating the class of the current CIM object. The __CLASS keyword in a WHERE clause only selects CIM instances of derived classes made out of the CLASS-NAME.
ASSOCIATORS OF	A keyword that is a WQL statement to locate associated CIM classes or CIM instances. It MUST NOT be used in combination with the SELECT keyword and the REFERENCES OF keyword.
OPTIONAL-ASSOC-WHERE	If the WHERE statement is specified in an ASSOCIATORS OF WQL query, it narrows the scope to one or several characteristics of the association and associated CIM classes. The filter expression can be made of several specific keywords and expressions to validate these characteristics. Each expression MUST be separated by a BLANK character, as specified in the ABNF notation above. Each expression MUST NOT be used more than once in a single WQL query. The keyword supported to narrow the scope of an ASSOCIATORS OF query are AssocClass, RequiredAssocQualifier, RequiredQualifier, ResultClass, ResultRole, Role, KeysOnly, and ClassDefsOnly.
REFERENCES OF	A keyword that is a WQL statement to locate the CIM classes or CIM instances associating CIM classes or CIM instances. It MUST NOT be used in combination with the SELECT keyword and the ASSOCIATORS OF keyword.
OPTIONAL-REF-WHERE	If the WHERE statement is specified in a REFERENCES OF query, it narrows the scope to one or several characteristics of the association and associated classes. The filter expression can be made of several specific keywords and expressions to express these characteristics. Each expression MUST be separated by a BLANK character. Each expression MUST NOT be used more than once in a single WQL query. The keyword supported to narrow the scope of an REFERENCES OF query are RequiredQualifier, ResultClass, Role, KeysOnly, and ClassDefsOnly.
OBJECT-REL-PATH	The **CIM relative path** of the CIM class or CIM instance to be queried. It MUST be specified for ASSOCIATORS OF and REFERENCES OF queries.
KeysOnly	If the KeysOnly keyword is being used in ASSOCIATORS OF and "REFERENCES OF" queries only the key properties of resulting CIM instances MUST be populated.
ClassDefsOnly	If the ClassDefsOnly keyword is being used in ASSOCIATORS OF and REFERENCES OF queries only the CIM class definitions of resulting CIM instances MUST be returned.
AssocClass	If the AssocClass keyword is being used in ASSOCIATORS OF queries, the resulting CIM instances MUST be associated with association class or CIM instances made out of the CLASS-NAME specified
RequiredAssocQualifier	If the RequiredAssocQualifier keyword is being used in ASSOCIATORS OF queries, the resulting CIM instances MUST have the CIM Qualifier of the given

Schema objects and keywords	Description
	name set in their association class or CIM instances.
RequiredQualifier	If the RequiredQualifier keyword is being used in ASSOCIATORS OF and REFERENCES OF queries, the resulting CIM instances MUST have the CIM Qualifier of the given name set.
ResultClass	If the ResultClass keyword is being used in ASSOCIATORS OF" and REFERENCES OF queries, the resulting CIM instances MUST be made out of the CLASS-NAME specified.
Role	If the Role keyword is being used in ASSOCIATORS OF and REFERENCES OF queries, the result MUST only return CIM instances where the role matches the reference CIM Property name of the association class.
ResultRole	If the ResultRole keyword is being used in ASSOCIATORS OF queries, the result MUST only return CIM instances where the role matches the reference CIM Property name of the CIM instances.

2.2.1.1.2 WQL Event Query

The following example shows the syntax for WQL event queries in Augmented BNF format.

```
; ------------------------------------
; WQL event queries
; ------------------------------------

EVENT-WQL = "SELECT" <PROPERTY-LIST> "FROM" /
              <EVENT-CLASS-NAME> <OPTIONAL-WITHIN> <EVENT-WHERE>

OPTIONAL-WITHIN = ["WITHIN" <INTERVAL>]
INTERVAL = 1*DIGIT
EVENT-WHERE = ["WHERE" <EVENT-EXPR>]

EVENT-EXPR = ( (<INSTANCE-STATE> "ISA" <CLASS-NAME> <EXPR2>) /
               <EXPR> )
               ["GROUP WITHIN" <INTERVAL>
                     ( ["BY" [<INSTANCE-STATE> DOT] <PROPERTY-NAME>]
                       ["HAVING" <EXPR>]] )
INSTANCE-STATE = "TARGETINSTANCE" / "PREVIOUSINSTANCE"

; ------------------------------------
; Expression
; ------------------------------------

EXPR =
 [OPEN-PARENTHESIS] <PROPERTY-EVALUATION> /
  <EXPR2> [CLOSE-PARENTHESIS]
EXPR2 = ( ["OR"  [OPEN-PARENTHESIS] <EXPR> /
   [CLOSE-PARENTHESIS] ] ) /
        ( ["AND" [OPEN-PARENTHESIS] <EXPR> /
   [CLOSE-PARENTHESIS] ] )

PROPERTY-EVALUATION =
 ( <PROPERTY-NAME> <OPERATOR> <TYPED-CONSTANT> ) /
  ( <PROPERTY-NAME> <IS-OPERATOR> "NULL" )
```

```
OPERATOR = <EQUIVALENT-OPERATOR> /
           <COMPARE-OPERATOR>

EQUIVALENT-OPERATOR = "=" / "!="
COMPARE-OPERATOR = "<=" / ">=" / "<" / ">" / "LIKE"
IS-OPERATOR = "IS" / "IS NOT"

; ------------------------------------
; Characters
; ------------------------------------

ALPHA = %x41-5A
DIGIT = %x30-39
DOT = ","
COMMA = "."
ASTERISK = "*"
OPEN-PARENTHESIS = "("
CLOSE-PARENTHESIS = ")"
STRING-IDENTIFIER = ALPHA *(ALPHA / DIGIT / (*("_") ALPHA / DIGIT))

CLASS-NAME = [__]<STRING-IDENTIFIER>
EVENT-CLASS-NAME = [__]<STRING-IDENTIFIER>
PROPERTY-NAME = [__]<STRING-IDENTIFIER>

TYPED-CONSTANT = INT /
                 REAL /
                 STRING /
                 DATETIME /
                 BOOL

INT = "[-+]?\d*"
REAL = "[-+]?\d*(\.\d+)?"
STRING = ["]([a-z][A-Z]\d)*["]
DATETIME = "(\d\d\d\d)(0\d|1[012])(0\d|[12][0-9]|3[01])
  ([0-1]\d|2[0-3])([0-5]\d)([0-5]\d)[.]\d\d\d\d\d\d[+-]
  ([0-6][02468][0]|7[0-2][0])"

BOOL = "TRUE" / "FALSE"
```

Objects and keywords	Description
EVENT-CLASS-NAME	Identifies an event CIM class name to be queried.
CLASS-NAME	Identifies a CIM class name to be queried for events.
PROPERTY-NAME	Identifies the name of a **CIM property** of a CIM class.
EVENT-WQL	A string expressing the WQL event query. The WQL string uses different WQL reserved keywords to select the type of information wanted.
SELECT	A keyword expressing the selection of information requested (similar to SQL SELECT). SELECT expresses the CIM class or CIM instance to be queried. It MUST be specified in a

Objects and keywords	Description
	WQL event query.
PROPERTY-LIST	A list of PROPERTY-NAME values. PROPERTY-NAME values in the list MUST be separated by a comma (",").
ASTERISK	Requires all properties of a CIM class or a CIM instance.
FROM	A keyword that MUST be specified with the SELECT statement to express the CIM class or CIM instance the query MUST be executed against.
EVENT-CLASS-NAME	MUST be specified and MUST be an intrinsic, an extrinsic, or a timer event class. An **intrinsic event** class is a class derived from __InstanceOperationEvent, __ClassOperationEvent, or __NamespaceOperationEvent, representing possible intrinsic events. An **extrinsic event** class is a class derived from __ExtrinsicEvent, representing possible extrinsic events. A timer event class is a class derived from __TimerEvent event class, representing possible timer events.
WITHIN	A keyword indicating the server to poll the system for an event. In case of an intrinsic EVENT-CLASS-NAME, the WITHIN keyword MUST be specified. The WITHIN keyword is optional for extrinsic EVENT-CLASS-NAME. If the WITHIN keyword is specified, the INTERVAL MUST be specified.
INTERVAL	INTERVAL specifies the polling interval. It MUST be expressed in seconds. If "WITHIN" is specified, the INTERVAL MUST be specified.
EVENT-WHERE	The WHERE statement narrows the scope of a SELECT event query if the EVENT-CLASS-NAME is an extrinsic or timer event CIM class. The WHERE statement MUST be specified to narrow the scope of a SELECT event query if the EVENT-CLASS-NAME is an intrinsic CIM class.
INSTANCE-STATE	Indicates the type of instance to be evaluated. INSTANCE-STATE MUST be specified if CLASS-NAME is an intrinsic CIM class. INSTANCE-STATE is optional if CLASS-NAME is an extrinsic CIM class. If specified, INSTANCE-STATE MUST be PREVIOUSINSTANCE - to indicate that the state of the CIM class or CIM instance before the event MUST be evaluated - or TARGETINSTANCE - to indicate that the state of the CIM class or CIM instance after the event MUST be evaluated.
ISA	A keyword that MUST be used in combination with the INSTANCE-STATE keyword. It is used as a comparative operator between the INSTANCE-STATE and a CLASS-NAME to reduce the scope of events returned to the CIM instances made out of the CLASS-NAME.
GROUP WITHIN	If the GROUP WITHIN keyword is used, the INTERVAL MUST be specified. This keyword indicates that all events occurring during the WITHIN INTERVAL period MUST be grouped as one event.
HAVING	If the HAVING keyword is specified, it MUST be followed by EXPR to filter the selection of events. This keyword indicates that all events grouped during the GROUP WITHIN period MUST meet the EXPR condition before being returned as one event.
BY	A keyword that groups event instances sharing a same value on a specified PROPERTY-NAME. In such a case, events representing a group of event sharing the same PROPERTY-NAME value is returned. The system MUST return as many events representing a group of events as there are PROPERTY-NAME values.
EVENT-EXPR	An expression for filtering WMI events.

2.2.1.2 CIM Path and Namespace

The syntax for CIM path and namespace is provided in Augmented BNF format.

```
; ----------------------------------
; CIM PATH
; ----------------------------------

CIMPATH = ( <NAMESPACE-PATH> COLON <OBJECT-PATH> ) /
  <OBJECT-PATH>
   NAMESPACE-PATH = [<MACHINE-PATH>] NAMESPACE
   MACHINE-PATH = BACKSLASH BACKSLASH <MACHINENAME> BACKSLASH
   OBJECT-PATH = <CLASS-NAME> [<INSTANCE-KEY>]
   INSTANCE-KEY = (EQUAL "@") / DOT <KEY-VALUE-LIST>
   KEY-VALUE-LIST = <PROPERTY-NAME> EQUAL
     <TYPED-CONSTANT> <KEY-VALUE-LIST2>
   KEY-VALUE-LIST2 = [ COMMA KEY-VALUE-LIST ]


CLASS-NAME = [__]<STRING-IDENTIFIER>
PROPERTY-NAME = [__]<STRING-IDENTIFIER>

; ----------------------------------
; NAMESPACE
; ----------------------------------

NAMESPACE = <STRING-IDENTIFIER> <SUB-NAMESPACE>
<SUB-NAMESPACE> = [ BACKSLASH <NAMESPACE> ]

TYPED-CONSTANT = INT /
                 REAL /
                 STRING /
                 DATETIME /
                 BOOL

INT = "[-+]?\d*"
REAL = "[-+]?\d*(\.\d+)?"
STRING = ["]([a-z][A-Z]\d)*["]
DATETIME =
"(\d\d\d\d)(0\d|1[012])(0\d|[12][0-9]|3[01])
  ([0-1]\d|2[0-3])([0-5]\d)([0-5]\d)[.]\d\d\d\d\d\d[+-]
  ([0-6][02468][0]|7[0-2][0])"

BOOL = "TRUE" / "FALSE"

; ----------------------------------
; Characters
; ----------------------------------

ALPHA = %x41-5A
DIGIT = %x30-39
BACKSLASH = "\"
DOT = "."
STRING-IDENTIFIER = ALPHA *(ALPHA / DIGIT / (*("_") ALPHA / DIGIT))
COLON=":"
MACHINENAME = <STRING-IDENTIFIER> / DOT
```

Objects and keywords	Description
OBJECT-PATH	The path of the CIM class or CIM instance to be referenced.
MACHINENAME	The network identifiable name of the machine where the WMI class, instance, or namespace referenced resides.
CLASS-NAME	Identifies a CIM class name.
INSTANCE-KEY	Uniquely identifies the instance of a given CIM class.
KEY-VALUE-LIST	List of PROPERTY-NAME and their values, separated by a ",". Each property value pair is represented as propertyName=value format.
PROPERTY-NAME	Identifies the name of a property of the CIM class.

2.2.1.3 Protocol Return Codes

Codes returned by the protocol are represented as HRESULT, as specified in [MS-ERREF] section 4.

The HRESULT values documented in the following table are interpreted by the protocol through a specific set of interfaces methods, as specified in sections 3.1.5.2, 3.1.5.3.2, and 3.1.5.5.1.

The severity flag MUST be interpreted, as specified in [MS-ERREF]. HRESULT errors are not recoverable by the protocol. HRESULT successes other than the ones specified in the following table MUST be treated equally as WBEM_S_NO_ERROR.

Constant/value	Description
WBEM_S_NO_ERROR 0x00000000	The operation was successful.
WBEM_S_FALSE 0x00000001	No more CIM objects are available, the number of CIM objects returned is less than the number requested, or this is the end of an enumeration. This value MUST also be returned when the enumeration methods are called with a value of 0 for the counting parameter (variable uCount or nCount in respect of the method definition), as specified in **IEnumWbemClassObject::Skip) (section 3.1.5.3.5)**, **IEnumWbemClassObject::Next (section 3.1.5.3.2)**, and **IWbemWCOSmartEnum::Next (section 3.1.5.7.1)**.
WBEM_S_TIMEDOUT 0x00040004	A call timed out. This is not an error condition.
WBEM_S_NEW_STYLE 0x000400FF	The operation was successful and indicates that the receiver of the call is able to receive Optimized **IWbemObjectSink::Indicate** calls.

2.2.1.4 IWbemClassObject

The signatures of many methods related to the Windows Management Instrumentation Remote Protocol include a parameter to specify an IWbemClassObject interface pointer. This parameter MUST be custom marshaled by the DCOM Remote Protocol, as specified below. The IWbemClassObject interface represents a WMI object, such as a WMI class or an object instance. All CIM object (CIM Classes and CIM Instances) passed during WMI calls between client and server are objects of this interface.

Parameter/ source	Value/description
Interface UUID	{DC12A681-737F-11CF-884D-00AA004B2E24}
Marshaling buffer layout	The buffer representing a CIM object MUST be encoded as specified in [MS-WMIO].
Unmarshaler CLSID	{4590F812-1D3A-11D0-891F-00AA004B2E24} This CLSID MUST represent the unmarshaler CLSID supplied by WMI to DCOM and MUST be sent over the network by DCOM when custom marshaling is implemented. For more information (OBJREF_CUSTOM), see the DCOM Remote Protocol.

2.2.1.5 WBEM_CHANGE_FLAG_TYPE Enumeration

The **WBEM_CHANGE_FLAG_TYPE** enumeration is used to indicate and update the type of the flag.

```
typedef [v1_enum] enum tag_WBEM_CHANGE_FLAG_TYPE
{
  WBEM_FLAG_UPDATE_ONLY = 0x01,
  WBEM_FLAG_CREATE_ONLY = 0x02,
  WBEM_FLAG_UPDATE_SAFE_MODE = 0x20,
  WBEM_FLAG_UPDATE_FORCE_MODE = 0x40
} WBEM_CHANGE_FLAG_TYPE;
```

WBEM_FLAG_UPDATE_ONLY: This flag causes the put operation to update the class or instance. The class or instance must exist for the call to be successful.

WBEM_FLAG_CREATE_ONLY: This flag causes the put operation to create the class or instance. The class or instance must NOT exist for the call to be successful.

WBEM_FLAG_UPDATE_SAFE_MODE: This flag allows updates of classes even if there are child classes, as long as the change does not cause any conflicts with child classes. An example of an update this flag would allow would be to add a new property to the base class that was not previously mentioned in any of the child classes. If the class has instances, the update fails.

WBEM_FLAG_UPDATE_FORCE_MODE: This flag forces updates of classes when conflicting child classes exist. An example of an update this flag would force is if a class qualifier were defined in a child class, and the base class tried to add the same qualifier that conflicted with the existing one. In force mode, this conflict is resolved by deleting the conflicting qualifier in the child class.

2.2.1.6 WBEM_GENERIC_FLAG_TYPE Enumeration

The **WBEM_GENERIC_FLAG_TYPE** enumeration is used to indicate and update the type of the flag.

```
typedef [v1_enum] enum tag_WBEM_GENERIC_FLAG_TYPE
{
  WBEM_FLAG_RETURN_IMMEDIATELY = 0x10,
  WBEM_FLAG_FORWARD_ONLY = 0x20,
  WBEM_FLAG_SEND_STATUS = 0x80,
  WBEM_FLAG_ENSURE_LOCATABLE = 0x100,
```

```
  WBEM_FLAG_DIRECT_READ = 0x200,
  WBEM_FLAG_USE_AMENDED_QUALIFIERS = 0x20000
} WBEM_GENERIC_FLAG_TYPE;
```

WBEM_FLAG_RETURN_IMMEDIATELY: This flag causes the synchronous call to return immediately without waiting for the operation to complete. The call result parameter contains the IWbemCallResult object by using the status of the operation that can be retrieved.

WBEM_FLAG_FORWARD_ONLY: This flag causes a forward-only enumerator (**IEnumWbemClassObject, (section 3.1.5.3)**) to be returned. Forward-only enumerators are generally much faster and use less memory than conventional enumerators but do not allow calls to **IEnumWbemClassObject::Clone** or **IEnumWbemClassObject::Reset**.

WBEM_FLAG_SEND_STATUS: This flag registers a request with WMI to receive intermediate status reports through the client's implementation of **IWbemObjectSink::SetStatus**, if supported by the provider.

WBEM_FLAG_ENSURE_LOCATABLE: This flag ensures that any returned objects have enough information in them so that the system properties, such as __PATH, __RELPATH, and __SERVER<1>, are non-NULL.

WBEM_FLAG_DIRECT_READ: This flag causes direct access to the provider for the class specified without any regard to its parent class or **subclasses**.

WBEM_FLAG_USE_AMENDED_QUALIFIERS: If this flag is set, WMI retrieves the amended qualifiers stored in the localized namespace of the current connection's locale. If not set, only the qualifiers stored in the immediate namespace are retrieved.

2.2.1.7 WBEM_STATUS_TYPE Enumeration

The **WBEM_STATUS_TYPE** enumeration gives information about the status of the operation.

```
typedef  enum tag_WBEM_STATUS_TYPE
{
  WBEM_STATUS_COMPLETE = 0,
  WBEM_STATUS_PROGRESS = 0x02
} WBEM_STATUS_TYPE;
```

WBEM_STATUS_COMPLETE: When the WMI operation is completed, WMI calls **IWbemObjectSink::SetStatus** with WBEM_STATUS_COMPLETE

WBEM_STATUS_PROGRESS: WMI reports the progress of the operation to **IWbemObjectSink::SetStatus** with flag WBEM_STATUS_PROGRESS

2.2.1.8 WBEM_TIMEOUT_TYPE Enumeration

The **WBEM_TIMEUT_TYPE** enumeration gives information about the type of timeout for the process.

```
typedef [v1_enum] enum tag_WBEM_TIMEOUT_TYPE
{
  WBEM_NO_WAIT = 0,
  WBEM_INFINITE = 0xFFFFFFFF
```

```
} WBEM_TIMEOUT_TYPE;
```

WBEM_NO_WAIT: If passed as a timeout parameter to **IEnumWbemClassObject::Next**, the call returns immediately whether or not objects are available.

WBEM_INFINITE: If passed as a timeout parameter to **IEnumWbemClassObject::Next**, the call blocks until objects are available.

2.2.1.9 WBEM_QUERY_FLAG_TYPE Enumeration

The **WBEM_QUERY_FLAG_TYPE** enumeration gives information about the type of the flag.

```
typedef [v1_enum] enum tag_WBEM_QUERY_FLAG_TYPE
{
  WBEM_FLAG_DEEP = 0,
  WBEM_FLAG_SHALLOW = 1,
  WBEM_FLAG_PROTOTYPE = 2
} WBEM_QUERY_FLAG_TYPE;
```

WBEM_FLAG_DEEP: If used in **IWbemServices::CreateClassEnum** or **IWbemServices::CreateClassEnumAsync**, it causes the enumeration to return all the subclasses in the hierarchy of a given class bt not the class itself.

If used in **IWbemServices::CreateInstanceEnum** or **IWbemServices::CreateInstanceEnumAsync**, it causes the enumeration to return the instances of this class, as well as instances of subclasses in the hierarchy of the class.

WBEM_FLAG_SHALLOW: If used in **IWbemServices::CreateClassEnum** or **IWbemServices::CreateClassEnumAsync**, it causes the enumeration to return the immediate subclasses of a given class.

If used in **IWbemServices::CreateInstanceEnum** or **IWbemServices::CreateInstanceEnumAsync**, it causes the enumeration to return only the instances of this class, excluding all instances of subclasses.

WBEM_FLAG_PROTOTYPE: This flag is used for prototyping. It does not execute the query but instead returns an object that looks like a typical result object.

2.2.1.10 WBEM_BACKUP_RESTORE_FLAGS Enumeration

The **WBEM_BACKUP_RESTORE_FLAGS** enumeration gives information about the backup and restore state of the process.

```
typedef [v1_enum] enum tag_WBEM_BACKUP_RESTORE_FLAGS
{
  WBEM_FLAG_BACKUP_RESTORE_FORCE_SHUTDOWN = 1
} WBEM_BACKUP_RESTORE_FLAGS;
```

WBEM_FLAG_BACKUP_RESTORE_FORCE_SHUTDOWN: While restoring the WMI repository, if any clients are connected to WMI, they are forcibly disconnected.

2.2.1.11 WBEMSTATUS Enumeration

The **WBEMSTATUS** enumeration gives information about the status of the operation.

```
typedef [v1_enum] enum tag_WBEMSTATUS
{
  WBEM_S_NO_ERROR = 0x00,
  WBEM_S_FALSE = 0x01,
  WBEM_S_TIMEDOUT = 0x40004,
  WBEM_E_FAILED = 0x80041001,
  WBEM_E_NOT_AVAILABLE = 0x80041009,
  WBEM_E_NOT_SUPPORTED = 0x8004100c,
  WBEM_E_INVALID_OPERATION = 0x80041016,
  E_NOTIMPL = 0x80004001
} WBEMSTATUS;
```

WBEM_S_NO_ERROR: The connection was established with no error.

WBEM_S_FALSE: Return a value of false regardless of status.

WBEM_S_TIMEDOUT: The attempt to establish the connection has expired.

WBEM_E_FAILED: The attempt to establish the connection has failed.

WBEM_E_NOT_AVAILABLE: The resource is not available.

WBEM_E_NOT_SUPPORTED: The operation attempted is not supported.

WBEM_E_INVALID_OPERATION: The operation attempted is not valid for this scenario.

E_NOTIMPL: The operation attempted is not implemented. The value of this element is as specified in [MS-ERREF] section 2.1.

2.2.1.12 WBEM_REFRESHER_FLAGS Enumeration

The **WBEM_REFRESHER_FLAGS** enumeration is used to indicate and update the type of the flag.

```
typedef [v1_enum] enum tag_WBEM_REFRESHER_FLAGS
{
  WBEM_FLAG_REFRESH_AUTO_RECONNECT = 0,
  WBEM_FLAG_REFRESH_NO_AUTO_RECONNECT = 1
} WBEM_REFRESHER_FLAGS;
```

WBEM_FLAG_REFRESH_AUTO_RECONNECT: This flag causes the refresher to reconnect to the provider, if the provider connection breaks.

WBEM_FLAG_REFRESH_NO_AUTO_RECONNECT: This flag causes the refresher not to reconnect to the provider, if the provider connection breaks.

2.2.1.13 WBEM_CONNECT_OPTIONS Enumeration

The **WBEM_CONNECT_OPTIONS** enumeration gives information about the type of options of the connection.

```
typedef [v1_enum] enum tag_WBEM_CONNECT_OPTIONS
{
  WBEM_FLAG_CONNECT_REPOSITORY_ONLY = 0x40,
  WBEM_FLAG_CONNECT_PROVIDERS = 0x100
} WBEM_CONNECT_OPTIONS;
```

WBEM_FLAG_CONNECT_REPOSITORY_ONLY: The connection is established to operate only on the static data (classes and instances) stored in the database. Operations requiring a provider will not be supported on this connection.

WBEM_FLAG_CONNECT_PROVIDERS: The connection is established to operate only on the provider.

2.2.1.14 IWbemContext

The signatures of many methods related to the Windows Management Instrumentation Remote Protocol include a parameter to specify an IWbemContext **interface pointer**. The IWbemContext interface represents an IWbemContext object<2>, acting as a property bag (specialized container for properties that stores Variants) that a client MAY use to store additional information to be used by the server. This information MUST be made of a property list, the property types, and the assigned property values.

When used through Windows Management Instrumentation Remote Protocol methods, the *IWbemContext* parameter MUST be custom marshaled by the DCOM Remote Protocol, as specified below.

Parameter/source	Value/description
Interface UUID	{44ACA674-E8FC-11D0-A07C-00C04FB68820}
Marshaling buffer layout	See the structures listed below.
Unmarshaler CLSID	{674B6698-EE92-11D0-AD71-00C04FD8FDFF} This CLSID represents the unmarshaler CLSID supplied by the Windows Management Instrumentation Remote Protocol to DCOM Remote Protocol and MUST be sent over the network by DCOM Remote Protocol when custom marshaling is implemented. Refer to [MS-DCOM], OBJREF_CUSTOM, section 2.2.1.16.6 for more information.

See Appendix A, MS-WMI Full **Idl** for the IDL of these two IWbemContext interfaces.

All scalar types encountered in the following structures MUST be stored in little-endian format.

Methods

This interface has no methods.

Structures

The **IWbemContext** interface defines the following structures.

Structure	Description
IWbemContext Buffer Marshaling	Structure requirements for marshaling a buffer.

Structure	Description
IWbemContext Property Marshaling	Structure requirements for marshaling a property.
IWbemContext String Marshaling	Structure requirements for marshaling a string.
IWbemContext Array Marshaling	Structure requirements for marshaling an array.

This interface inherits the IUnknown interface. Method **opnum** field values start with 3; opnum values 0–2 represent the IUnknown_QueryInterface, IUnknown_AddRef, and IUnknown_Release methods, respectively, as specified in [MS-DCOM].

2.2.1.14.1 IWbemContextBuffer

The stream MUST start with a 32-bit integer (*numGUIDs* below) which MUST be set to 0.

Next, the stream MUST contain a 32-bit integer representing the property count. The property list MUST immediately follow the property count and MUST be marshaled as a continuous list without padding between properties.

IWbemContextBuffer has the following structure.

```
typedef struct {
  UINT32 numGUIDs;
  UINT32 numProps;
  byte props[];
} IWbemContextBuffer;
```

numGUIDs: MUST always be set to 0.

numProps: MUST represent the number of properties contained in the **IWbemContext** structure.

props: Variable-length field which MUST contain the marshaled properties. Each property MUST be marshaled, as specified in section 2.2.1.14.2.

2.2.1.14.2 IWbemContextProperty

The property is a variable-length structure. It MUST conform to the following network representation.

IWbemContextProperty has the following structure.

```
typedef struct {
  IWbemContextString propName;
  UINT32 lFlags;
  UINT32 variantType;
  byte propValue[];
} IWbemContextProperty;
```

propName: The property name MUST be marshaled as a string in the format specified in 2.2.1.14.3.

lFlags: MUST be set to 0 and ignored.

variantType: A representing the property type, MUST have one of the following values as specified in [MS-OAUT], section **2.2.4**:

- VT_NULL
- VT_I2
- VT_I4
- VT_R4
- VT_R8
- VT_BSTR
- VT_BOOL
- VT_UI1
- VT_UI2
- VT_UI4
- VT_UNKNOWN
- VT_I1.

If the value is an array, the types above MUST combined by using the bitwise OR operation with VT_ARRAY (also specified in [MS-OAUT] section **2.2.4**).

propValue: Variable-length field that MUST contain the marshaled value of the property.

This property value is marshaled as given in the following table:

Property types	Marshaling
VT_BSTR	MUST be marshaled as an IWbemContextString.
VT_IUNKNOWN	MUST be marshaled as a buffer for IWbemClassObject interface.
VT_NULL	MUST be marshaled as an array of size 0.
VT_UI1, VT_I1.	MUST be marshaled as an array of 8 bytes with the first byte containing the value of the property.
VT_I2, VT_UI2, VT_BOOL,	MUST be marshaled as an array of 8 bytes with the first 2 bytes containing the value of property.
VT_I4, VT_UI4	MUST be marshaled as an array of 8 bytes with the first 4 bytes containing the value of property.
VT_R4	MUST be marshaled as an array of 8 bytes with the first 4 bytes containing the value of the property, as specified in [IEEE754], a 4-byte floating-point format.
VT_R8	MUST be marshaled as specified in [IEEE754], an 8-byte floating-point format.
VT_ARRAY | VT_*	MUST be marshaled as an array format as specified in 2.2.1.14.4.

2.2.1.14.3 IWbemContextString

Strings (property names and VT_BSTR properties values) MUST be represented as a 32-bit character count followed by the characters in UTF-16, as specified in [UNICODE].

IWbemContextString has the following structure.

```
typedef struct {
  UINT32 length;
  byte name[];
} IWbemContextString;
```

length: MUST represent the string length.

name: A variable-length field that MUST contain a stream of (length) UTF-16 characters.

2.2.1.14.4 IWbemContextArray

IWbemContextArray has the following structure.

```
typedef struct {
  UINT32 elCount;
  UINT32 ElSize;
  byte Element[];
} IWbemContextArray;
```

elCount: MUST be an integer representing the array's element count.

ElSize: MUST represent the array's element size. The size MUST match the size specified below.

Element: A variable stream of bytes representing all elements in the array.

Each element MUST be marshaled as an array of bytes using the following representation (array elements are marshaled in a different representation than non-array elements).

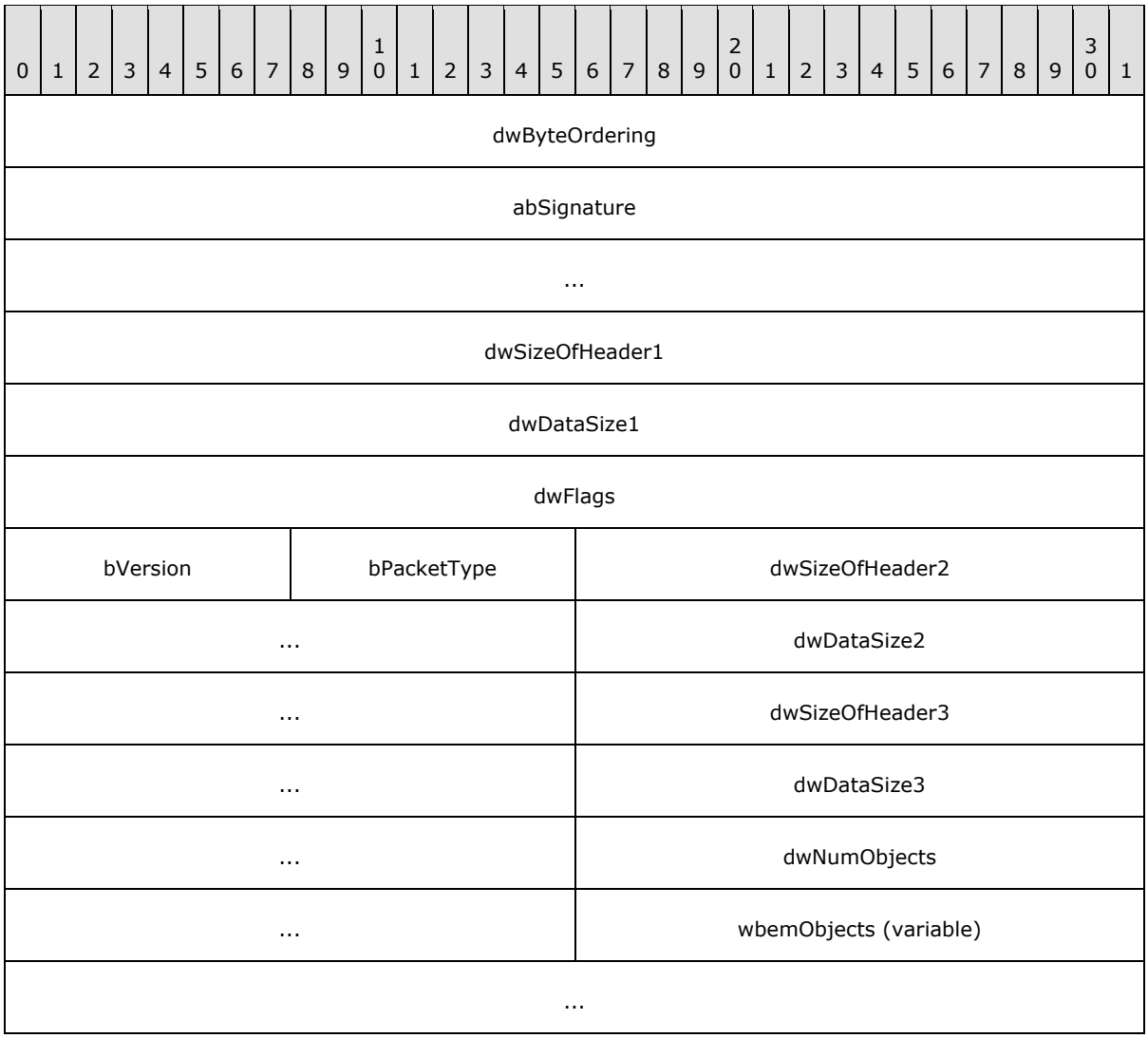
Type	Marshaling
VT_BSTR	MUST be marshaled as an **IWbemContextString**.
VT_IUNKNOWN	MUST be marshaled as an array of bytes representing a marshaling buffer for IWbemClassObject interface.
VT_NULL	MUST be marshaled as 0 bytes.
VT_I1, VT_UI1	MUST be marshaled as 1 byte.
VT_I4, VT_UI4	MUST be marshaled as 4-byte little-endian format.
VT_R4	MUST be marshaled as an array of 8 bytes with the first 4 bytes containing the value of the property, as specified in [IEEE754], a 4-byte floating-point format.
VT_R8	MUST be marshaled as specified in [IEEE754], an 8-byte floating-point format.

Type	Marshaling
VT_I2, VT_BOOL, VT_UI2	MUST be marshaled as an 2-byte little-endian format.

2.2.1.15 ObjectArray

The ObjectArray structure MUST be used to encode multiple CIM objects returned in response to the **IWbemWCOSmartEnum::Next (section 3.1.5.7.1)** method. This structure is also used to encode parameters of the optimized **IWbemObjectSink::Indicate (section 3.1.5.5.1)** method.<3> To minimize network bandwidth, a server SHOULD support the ObjectArray structure when sending an array of CIM objects.

The optimization MUST be achieved by sending the CIM class information just once at the beginning of the communication. This CIM class MUST be identified by a randomly generated GUID maintained by both the server and the client. The remaining CIM instances MUST be sent without the CIM class class information. The CIM class definition identified by the **GUID** is used to reconstruct the full CIM instances on the client side.



dwByteOrdering (4 bytes): The byte ordering. MUST be one of the following values.<4>

Value	Meaning
0x00000000	Value when byte ordering is little-endian.
0xFFFFFFFF	Value when byte ordering is big-endian.

abSignature (8 bytes): MUST be set to {0x57, 0x42, 0x45, 0x4D, 0x44, 0x41, 0x54, 0x41} (a byte array containing the unquoted, unterminated ASCII string "WBEMDATA").

dwSizeOfHeader1 (4 bytes): This stores the total size of these fields: **dwByteOrdering**, **abSignature**, **dwSizeofHeader1**, **dwDataSize1**, **dwFlags**, **bVersion**, and **bPacketType**.

The size of the header MUST be 0x0000001A. Data immediately follows the header.

dwDataSize1 (4 bytes): MUST indicate the length of the data following this header in bytes, starting at the **dwSizeOfHeader2** field.

dwFlags (4 bytes): The flag value MUST be 0x00000000.

bVersion (1 byte): The version number of the header. The version MUST be 1.

bPacketType (1 byte): The value of this field is dependent on the call context.

Value	Meaning
0x00000000	Value in the context of an optimized **IWbemObjectSink::Indicate** (section 3.1.5.5.1) call.
0x00000001	Value in the context of an optimized **IWbemWCOSmartEnum::Next** (section 3.1.5.7.1) call.

dwSizeOfHeader2 (4 bytes): This stores the size of these fields: **dwSizeofHeader2** and **dwDataSize2**.

This size MUST be 8. Data immediately follows after the field **dwDataSize2**.

dwDataSize2 (4 bytes): MUST be the size in bytes of the data following this field.

dwSizeOfHeader3 (4 bytes): This stores the size of these fields: **dwSizeofHeader3**, **dwDataSize3**, and **dwNumObjects**. This size MUST be 12. Data immediately follows after the field **dwNumObjects**.

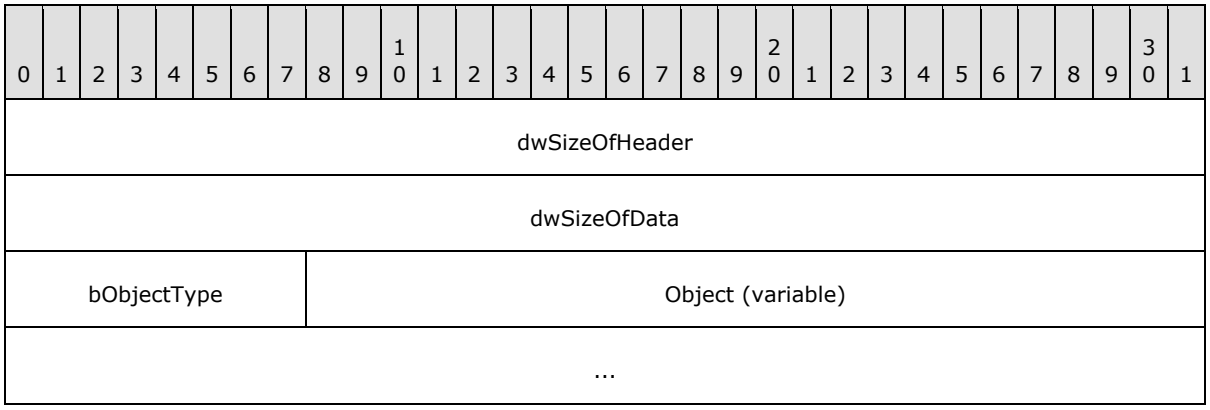
dwDataSize3 (4 bytes): MUST indicate the length of the remaining data, starting at the **wbemObjects** field.

dwNumObjects (4 bytes): MUST be the number of CIM objects in the ObjectArray.

wbemObjects (variable): The objects array containing the CIM class definition and CIM instances. These CIM objects MUST be encoded in the WBEM_DATAPACKET_OBJECT structure.

2.2.1.15.1 WBEM_DATAPACKET_OBJECT Structure

The WBEM_DATAPACKET_OBJECT MUST contain the CIM class definition and CIM instances.



dwSizeOfHeader (4 bytes): The size, in bytes, of the WBEM_DATAPACKET_OBJECT header, which MUST be 0x00000009.

dwSizeOfData (4 bytes): The size, in bytes, of the data following the WBEM_DATAPACKET_OBJECT header.

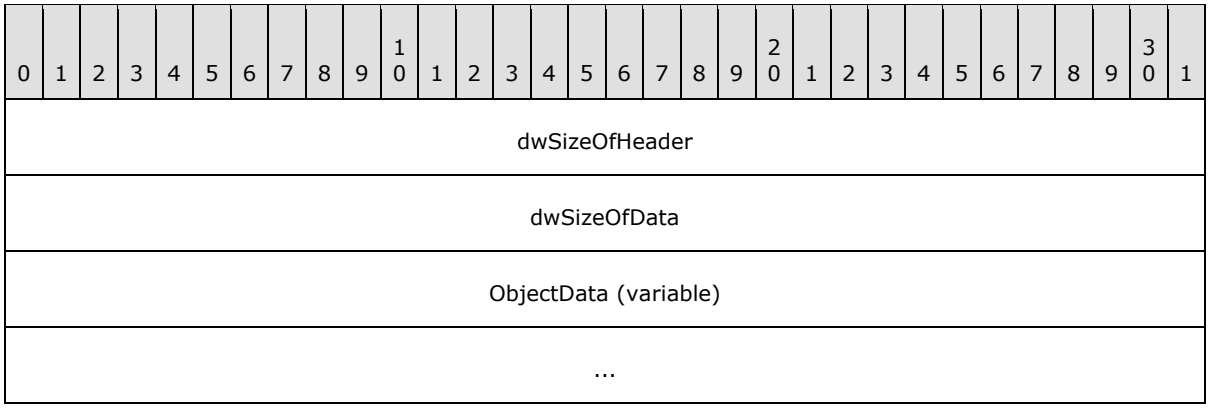
bObjectType (1 byte): The type of data in the data packet. The type MUST take one of the following specified values.

Value	Meaning
1	Object is type WBEMOBJECT_CLASS Structure contains the complete CIM Class definition.
2	Object is type WBEMOBJECT_INSTANCE Structure contains the complete CIM Instance definition.
3	Object is type WBEMOBJECT_INSTANCE_NOCLASS Structure contains CIM Instance without the CIM Class definition.

Object (variable): The CIM object carried into the WBEM_DATAPACKET_OBJECT, having dwSizeOfData bytes. The embedded CIM object MUST match the selector field **bObjectType**.

2.2.1.15.2 WBEMOBJECT_CLASS Structure

The WBEMOBJECT_CLASS structure MUST contain a complete CIM class definition.



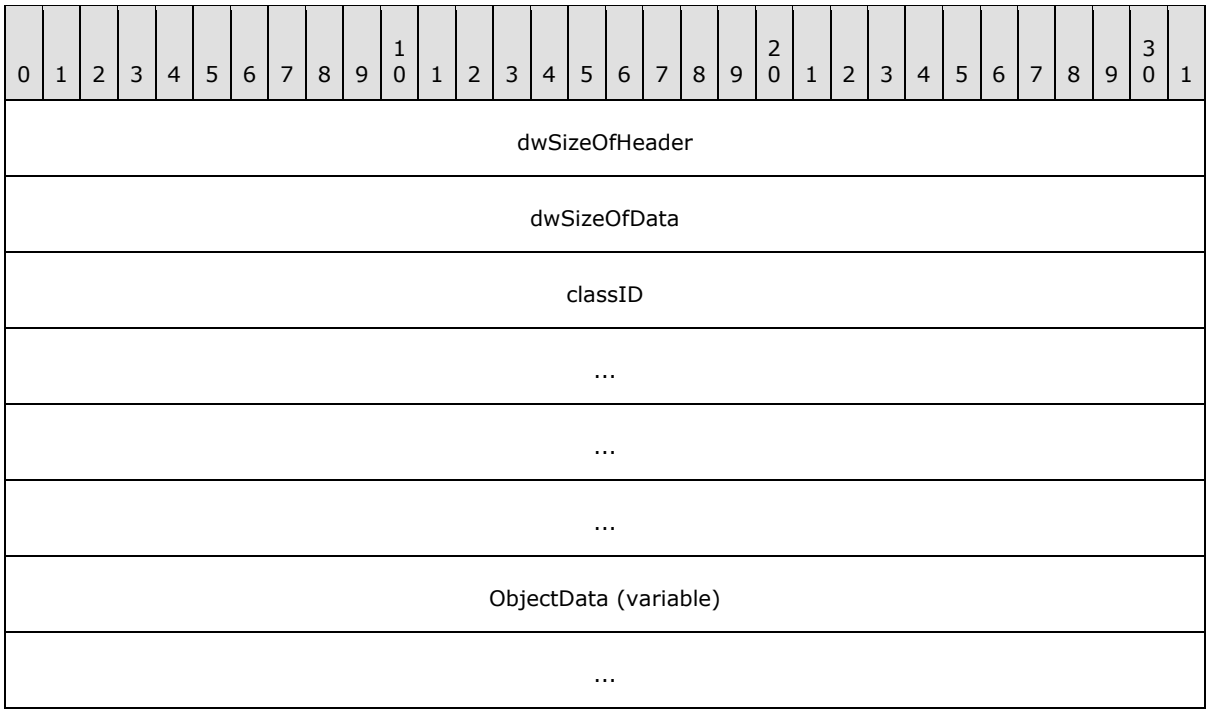
dwSizeOfHeader (4 bytes): The size, in bytes, of the header, which MUST be 0x00000008.

dwSizeOfData (4 bytes): The size, in bytes, of the data following the header.

ObjectData (variable): The string of bytes (ObjectBlock, as specified in [MS-WMIO] section **2.2.3**) of **dwSizeOfData** length, representing the marshaled buffer of a IWbemClassObject associated with a CIM class.

2.2.1.15.3 WBEMOBJECT_INSTANCE Structure

The WBEMOBJECT_INSTANCE structure MUST contain a complete CIM instance.



dwSizeOfHeader (4 bytes): The size, in bytes, of the header, which MUST be 0x00000018.

dwSizeOfData (4 bytes): The size, in bytes, of the data following the header.

classID (16 bytes): The unique identifier of the CIM class type.

ObjectData (variable): The string of bytes (ObjectBlock, as specified in [MS-WMIO] section **2.2.3**) of **dwSizeOfData** length, representing the marshaled buffer of a IWbemClassObject associated with a CIM class.

2.2.1.15.4 WBEMOBJECT_INSTANCE_NOCLASS Structure

The WBEMOBJECT_INSTANCE_NOCLASS structure MUST contain a CIM instance without the CIM class definition.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
dwSizeOfHeader																															
dwSizeOfData																															
classID																															
...																															
...																															
...																															
ObjectData (variable)																															
...																															

dwSizeOfHeader (4 bytes): The size, in bytes, of the header, which MUST be 0x00000018.

dwSizeOfData (4 bytes): The size, in bytes, of the data following the header.

classID (16 bytes): The unique identifier of the CIM class type.

ObjectData (variable): MUST represent a IWbemClassObject as a sequence of the following elements constituting the IWbemClassObject, as specified in the respective sections of [MS-WMIO].

- ObjectFlags
- Decoration
- EncodingLength
- InstanceFlags
- InstanceClassName
- NdTable
- InstanceData
- InstanceQualifierSet
- InstanceHeap

2.2.1.16 WBEM_REFRESHED_OBJECT Structure

The **WBEM_REFRESHED_OBJECT** structure SHOULD be used to encode the results of remote refreshing service returned by the **IWbemRemoteRefresher::RemoteRefresh (section 3.1.5.13.1)** interface method.

```
typedef struct _WBEM_REFRESHED_OBJECT {
  long m_lRequestId;
  WBEM_INSTANCE_BLOB_TYPE m_lBlobType;
  long m_lBlobLength;
  [size_is(m_lBlobLength)] byte* m_pBlob;
} WBEM_REFRESHED_OBJECT;
```

m_lRequestId: MUST contain the request ID.

m_lBlobType: MUST represent the type of the CIM object encoded in m_pbBlob.

m_lBlobLength: MUST represent the length of the m_pbBlob array.

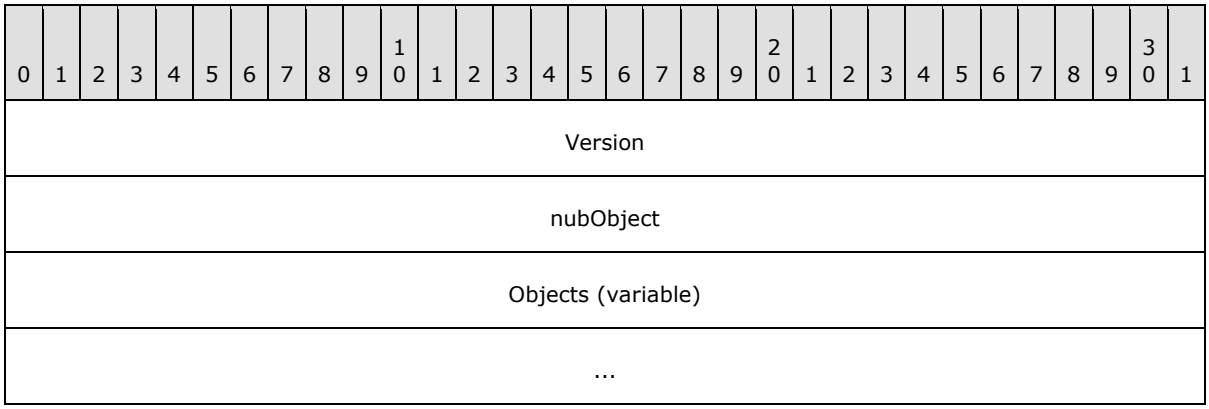
m_pBlob: When m_lBlobType is set to WBEM_BLOB_TYPE_ALL, it MUST contain the instance information represented in the WBEM_INSTANCE_BLOB format for a single IWbemClassObject interface pointer being part of the refreshing result. m_pbBlob is a byte stream representing a IWbemClassObject as a sequence of the following elements constituting the IWbemClassObject: InstanceData, InstanceQualifierSet, and InstanceHeap, as specified in [MS-WMIO].

When m_lBlobType is set to WBEM_BLOB_TYPE_ERROR, the m_lBlobLength MUST be set to 0.

When m_lBlobType is set to WBEM_BLOB_TYPE_ENUM, it MUST contain the instance information represented in the WBEM_INSTANCE_BLOB format for several IWbemClassObject interface pointers being part of the refreshing result.

2.2.1.17 WBEM_INSTANCE_BLOB

The WBEM_INSTANCE_BLOB is used to represent the refreshed object or enumeration in m_pBlob attribute of **WBEM_REFRESHED_OBJECT** structure.



Version (4 bytes): MUST represent the encoding version. Version MUST be set to 0x00000001.

nubObject (4 bytes): MUST represent the number of CIM objects encoded that are contained in the package.

Objects (variable): MUST contain the objects represented in RefreshedInstances format.

2.2.1.18 WBEM_INSTANCE_BLOB_TYPE

The **WBEM_INSTANCE_BLOB_TYPE** enumeration used to indicate the type of a CIM Object.

```
typedef  enum _WBEM_INSTANCE_BLOB_TYPE
{
  WBEM_BLOB_TYPE_ALL = 2,
  WBEM_BLOB_TYPE_ERROR = 3,
  WBEM_BLOB_TYPE_ENUM = 4
} WBEM_INSTANCE_BLOB_TYPE;
```

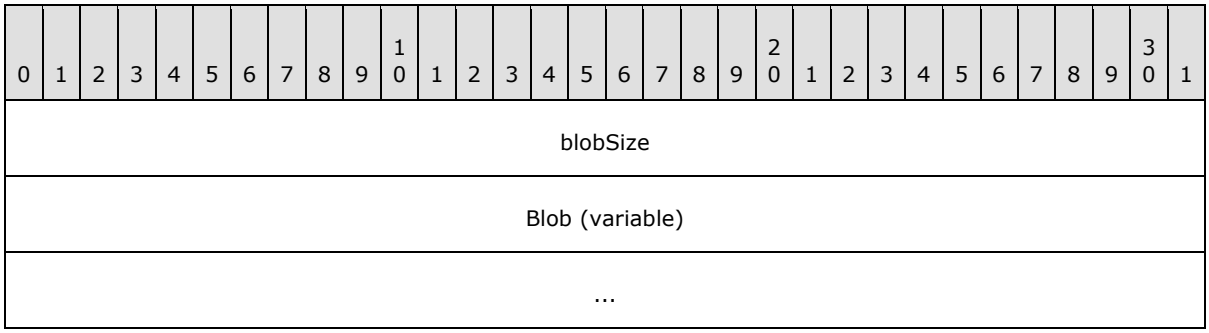
WBEM_BLOB_TYPE_ALL: The object is a single CIM object.

WBEM_BLOB_TYPE_ERROR: Represents an error condition. In this case the object is NULL.

WBEM_BLOB_TYPE_ENUM: The object is an enumeration of objects of a specific CIM type.

2.2.1.19 RefreshedInstances

The RefreshedInstances packet is contained within the WBEM_INSTANCE_BLOB.



blobSize (4 bytes): MUST represent the length of the blob array.

Blob (variable): MUST be a byte stream representing a partial IWbemClassObject as a sequence of the following elements constituting the original IWbemClassObject: InstanceData, InstanceQualifierSet, and InstanceHeap, as specified in [MS-WMIO].

2.2.1.20 _WBEM_REFRESH_INFO Structure

The **_WBEM_REFRESH_INFO** structure MUST be populated by the Windows Management Instrumentation Remote Protocol service providing the refresher information. The structure MUST be used to return to information from **IWbemRefreshingServices (section 3.1.5.12)** interface methods.

```
typedef struct {
  long m_lType;
  [switch_is(m_lType)] WBEM_REFRESH_INFO_UNION m_Info;
```

```
  long m_lCancelId;
} _WBEM_REFRESH_INFO;
```

m_lType: MUST be one of the constants specified in **WBEM_REFRESH_TYPE**.

m_Info: MUST be one of the WBEM_REFRESH_INFO_UNION types.

m_lCancelId: MUST be a unique identifier for the refresher object being used to cancel the refreshing object when using the **IWbemRemoteRefresher::StopRefreshing (section 3.1.5.13.2)**.

2.2.1.21 _WBEM_REFRESHER_ID Structure

The **_WBEM_REFRESHER_ID** structure identifies the client that is requesting refreshing services. The structure MUST be used to return information from **IWbemRefreshingServices (section 3.1.5.12)** interface methods.

```
typedef struct {
  [string] LPSTR m_szMachineName;
  DWORD m_dwProcessId ;
  GUID m_guidRefresherId;
} _WBEM_REFRESHER_ID;
```

m_szMachineName: MUST be the NetBIOS name of the client machine.

m_dwProcessId : It MUST be an identifier created by the client and it MUST be unique within the context of the client.<5>

m_guidRefresherId: MUST be a client-generated GUID.

2.2.1.22 _WBEM_RECONNECT_INFO Structure

The **_WBEM_RECONNECT_INFO** structure MUST contain the type for the information about the target CIM instance.

```
typedef struct {
  long m_lType;
  [string] LPCWSTR m_pwcsPath ;
} _WBEM_RECONNECT_INFO;
```

m_lType: MUST be one of the **WBEM_RECONNECT_TYPE** enumeration values.

m_pwcsPath : MUST be a **CIM path** to the remote CIM instance to be added to the refresher.

2.2.1.23 _WBEM_RECONNECT_RESULTS Structure

The **_WBEM_RECONNECT_RESULTS** structure defines the status of a reconnect operation. The structure MUST be used to return information from **IWbemRefreshingServices (section 3.1.5.12)** interface methods.

```
typedef struct {
  long m_lId;
  HRESULT m_hr;
} _WBEM_RECONNECT_RESULTS;
```

m_lId: MUST be a unique identifier for the refresher object used to cancel the refreshing object by using the IWbemRemoteRefresher::StopRefreshing (section 3.1.5.13.2) interface method.

m_hr: MUST be the HRESULT of the reconnect operation.

2.2.1.24 _WBEM_RECONNECT_TYPE Enumeration

The **_WBEM_RECONNECT_TYPE** enumeration defines possible types of remote CIM instances. The structure MUST be used to return to information from **IWbemRefreshingServices (section 3.1.5.12)** interface methods.

```
typedef  enum
{
  WBEM_RECONNECT_TYPE_OBJECT = 0,
  WBEM_RECONNECT_TYPE_ENUM = 1,
  WBEM_RECONNECT_TYPE_LAST = 2
} WBEM_RECONNECT_TYPE;
```

WBEM_RECONNECT_TYPE_OBJECT: The refresher should connect to refresh an object.

WBEM_RECONNECT_TYPE_ENUM: The refresher should connect to refresh an enumeration.

WBEM_RECONNECT_TYPE_LAST: This member is used only by the server to track the range of values for this enumeration. It MUST NOT be used by the client.

2.2.1.25 _WBEM_REFRESH_TYPE Enumeration

The **WBEM_REFRESH_TYPE** enumeration defines refresh types for the **_WBEM_REFRESH_INFO** structure.

```
typedef  enum
{
  WBEM_REFRESH_TYPE_INVALID = 0,
  WBEM_REFRESH_TYPE_REMOTE = 3,
  WBEM_REFRESH_TYPE_NON_HIPERF = 6
} _WBEM_REFRESH_TYPE;
```

WBEM_REFRESH_TYPE_INVALID: The server uses the value internally. The server MUST NOT return the value.

WBEM_REFRESH_TYPE_REMOTE: When the instance to be refreshed is derived from CIM_StatisticalInformation, the server MUST return WBEM_REFRESH_TYPE_REMOTE. In this case, m_Info contains the _WBEM_REFRESH_INFO_REMOTE structure.

WBEM_REFRESH_TYPE_NON_HIPERF: When the instance to be refreshed is NOT derived from CIM_StatisticalInformation, the server MUST return

WBEM_REFRESH_TYPE_NON_HIPERF. In this case, m_Info contains _WBEM_REFRESH_INFO_NON_HIPERF.

2.2.1.26 _WBEM_REFRESH_INFO_NON_HIPERF Structure

The **_WBEM_REFRESH_INFO_NON_HIPERF** structure MUST be returned by the server when the requested CIM instance cannot be part of the refreshing results.

```
typedef struct {
  [string] WCHAR* m_wszNamespace;
  IWbemClassObject* m_pTemplate;
} _WBEM_REFRESH_INFO_NON_HIPERF;
```

m_wszNamespace: MUST be a **CIM namespace** where enumeration of a given class exists.

m_pTemplate: MUST be a pointer to an IWbemClassObject interface, which MUST represent a CIM instance with all properties set to the default values. Default property values are as specified in [MS-WMIO] section 2.2.24.

2.2.1.27 _WBEM_REFRESH_INFO_REMOTE Structure

The **_WBEM_REFRESH_INFO_REMOTE** structure MUST be used when the client is on a different computer than the WMI service providing the refreshed information.

```
typedef struct {
  IWbemRemoteRefresher* m_pRefresher;
  IWbemClassObject* m_pTemplate;
  GUID m_Guid;
} _WBEM_REFRESH_INFO_REMOTE;
```

m_pRefresher: MUST be a pointer to the IWbemRemoteRefresher that the client used to retrieve an refreshed information.

m_pTemplate: MUST be a pointer to an IWbemClassObject interface that MUST represent a CIM instance with all properties set to the default values as specified in [MS-WMIO] section 2.2.24.

m_Guid: MUST be a globally unique identifier (GUID) created to identify this **_WBEM_REFRESH_INFO** object.

2.2.1.28 _WBEM_REFRESH_INFO_UNION Union

The **_WBEM_REFRESH_INFO_UNION** union defines a union of one of the following types: m_Remote, m_NonHiPerf, or m_hres.

```
typedef
[switch_type(long)]
union _WBEM_REFRESH_INFO_UNION {
  [case(WBEM_REFRESH_TYPE_REMOTE)]
    _WBEM_REFRESH_INFO_REMOTE m_Remote;
  [case(WBEM_REFRESH_TYPE_NON_HIPERF)]
```

```
    _WBEM_REFRESH_INFO_NON_HIPERF m_NonHiPerf;
  [case(WBEM_REFRESH_TYPE_INVALID)]
    HRESULT m_hres;
} WBEM_REFRESH_INFO_UNION;
```

m_Remote: An m_Remote _WBEM_REFRESH_INFO_REMOTE type.

m_NonHiPerf: An m_NonHiPerf _WBEM_REFRESH_INFO_NON_HIPERF type.

m_hres: An m_hres HRESULT type.

3 Protocol Details

The following sections specify details of the Windows Management Instrumentation Remote Protocol, including abstract data models, interface method syntax, and message processing rules.

3.1 Client and Server Details

A client in the context of this specification is a machine issuing a Windows Management Instrumentation Remote Protocol request. The request is issued against a Windows Management Instrumentation Remote Protocol server. In this context, a server is a machine handling the request issued by the client. Client requests MAY be issued **synchronously**, **semisynchronously**, or asynchronously. The detailed sequence diagrams for each of these modes are as specified in section 4. However, an overview of a typical protocol sequence is illustrated below.

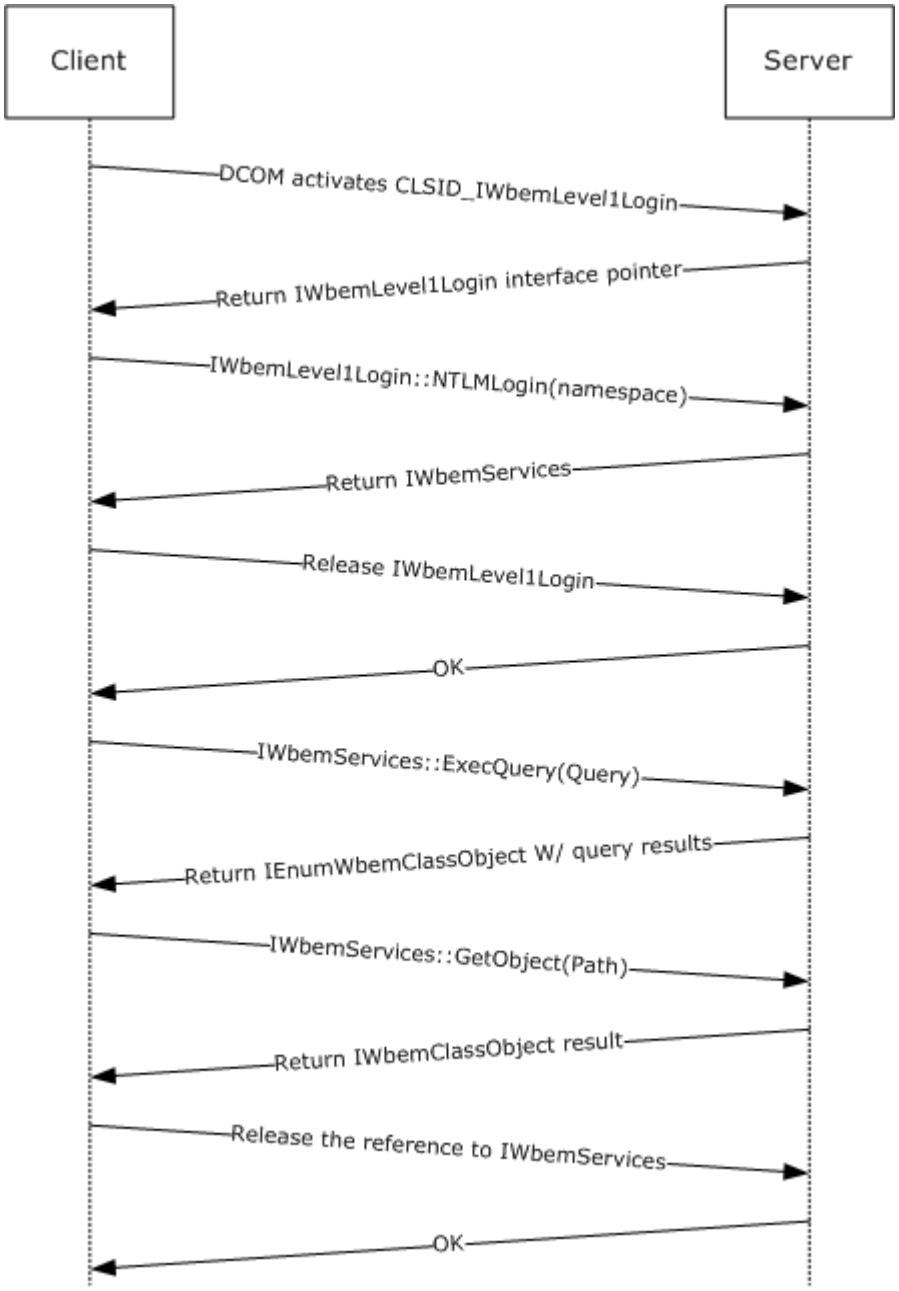


Figure 3: Typical protocol sequence

3.1.1 Abstract Data Model

The namespace implemented by a server is backed up by a CIM database and MAY be changed via the Windows Management Instrumentation Remote Protocol interfaces specified in this specification.

All dynamic data required for the protocol operation is stored in DCOM objects that are activated during the normal course of the protocol.

3.1.1.1 Call Sequences for Synchronous and Semisynchronous Operations Returning Multiple Objects

The **IEnumWbemClassObject** interface MUST be used to retrieve the results of **IWbemServices** interface methods returning an IWbemClassObject interface pointer.

An object of type **IEnumWbemClassObject** MUST be returned by the following methods:

- **IWbemServices::ExecQuery**
- **IWbemServices::CreateInstanceEnum**
- **IWbemServices::CreateClassEnum**
- **IWbemServices::ExecNotificationQuery**
- **IEnumWbemClassObject::Clone**

The client must call the **IEnumWbemClassObject::Next (section 3.1.5.3.2)** method repeatedly to retrieve the objects.

3.1.1.2 Call Sequences for Semisynchronous Operation Returning Single CIM Object

IWbemCallResult interface MUST be used to retrieve the result of **IWbemServices** interface methods returning an **IWbemCallResult** interface pointer.

The client MUST use the returned **IWbemCallResult** interface pointer to determine the status of a semisynchronous call as well as to retrieve the operation results.

The server MUST indicate the completion of the operation by returning WBEM_S_NO_ERROR in response to **IWbemCallResult::GetCallStatus**. The server MUST return the real results in response to other methods of **IWbemCallResult** interface depending of the original **IWbemServices** method call as follows.

Method	Rule
OpenNamespace	The **IWbemCallResult::GetResultService** method MUST be called to retrieve the new **IWbemServices** pointer.
PutInstance	The **IWbemCallResult::GetResultString** method MUST be called to obtain the CIM path that was assigned to the CIM instance.
GetObject	The **IWbemCallResult::GetResultObject** method MUST be called to retrieve the CIM object.
PutClass DeleteClass DeleteInstance	The **IWbemCallResult::GetCallStatus** method MUST be called to return the call status.
ExecMethod	The **IWbemCallResult::GetResultObject** method MUST be called to retrieve the output parameters.

3.1.1.3 Call Sequences for asynchronous Operation

To make an asynchronous query, the client MUST use **IWbemServices** methods providing asynchronous behavior. The methods providing asynchronous behavior are specified in the following sections:

- **IWbemServices::GetObjectAsync (section 3.1.5.2.5)**
- **IWbemServices::PutClassAsync (section 3.1.5.2.7)**
- **IWbemServices::DeleteClassAsync (section 3.1.5.2.9)**
- **IWbemServices::CreateClassEnumAsync (section 3.1.5.2.11)**
- **IWbemServices::PutInstanceAsync (section 3.1.5.2.13)**
- **IWbemServices::DeleteInstanceAsync (section 3.1.5.2.15)**
- **IWbemServices::CreateInstanceEnumAsync (section 3.1.5.2.17)**
- **IWbemServices::ExecQueryAsync (section 3.1.5.2.19)**
- **IWbemServices::ExecNotificationQueryAsync (section 3.1.5.2.21)**
- **IWbemServices::ExecMethodAsync (section 3.1.5.2.23)**

The client MUST implement an object exposing the **IWbemObjectSink Interface**, which MUST be passed in pResponseHandler parameter of the asynchronous method. The server MUST invoke the **IWbemObjectSink::Indicate** method to pass a new set of IWbemClassObject interface pointers, part of the operation result.

The server calls **IWbemObjectSink::Indicate** multiple times until the entire result set is delivered to the client. If no results are reported, the server MUST NOT call this method.

After the result set is delivered to the client, the server MUST call **IWbemObjectSink::SetStatus** to indicate the completion status of the asynchronous operation. The server MUST **release** the supplied **IWbemObjectSink** interface pointer.

Calls made by the server into the client provided **IWbemObjectSink** MAY be made at any authentication level.<6> The server MUST try to make the calls by using the machine principal name.

An overview of the call sequences for an asynchronous operation is there in sections 4.5 and 4.4.

3.1.2 Timers

The server MUST use timers to ensure that the conversation between itself and its clients remains active. The Windows Management Instrumentation Remote Protocol uses two timers:

- Sink timer: Each asynchronous operation has a corresponding timer, which MUST be initialized to 30 seconds when the server calls back the client using **IWbemObjectSink** methods. The timer MUST be reset when the call completes.
- Backup timer: Each **IWbemBackupRestoreEx** has a corresponding timer, which MUST be initialized to 15 minutes when the server receives a **IWbemBackupRestoreEx::Pause**. The timer MUST be reset when the server receives a **IWbemBackupRestoreEx::Resume**.

3.1.3 Initialization

The protocol MUST be initialized after successful activation of one of the two interfaces registered with the DCOM Remote Protocol infrastructure, as specified in [MS-DCOM] section 1.9.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

The server MUST accept multiple parallel invocations from different clients running under different **security principals**. On each interface, the server MUST support multiple outstanding calls.

The errors returned by the server are not actionable unless explicitly specified in this section. The server MUST perform an access check against all operations and ensures the secure access to the results.

The methods MUST be secured by using access rights as specified in section 5.2. Each method lists its own access right requirements.

All **IWbemServices** method calls MUST be executed on the interface obtained in responses to a previous call to **IWbemLevel1Login::NTLMLogin**, as specified in section 3.1.5.1.4, or to the **IWbemServices::OpenNamespace** method, as specified in section 3.1.5.2.1.

3.1.5.1 IWbemLevel1Login Interface

The **IWbemLevel1Login** interface allows a user to connect to the management services interface in a particular namespace. The interface MUST be implemented by the server. The interface MUST be uniquely identified by UUID F309AD18-D86A-11d0-A075-00C04FB68820.

Methods in RPC Opnum Order

Method	Description
EstablishPosition	Opnum: 3
RequestChallenge	Opnum: 4
WBEMLogin	Opnum: 5
NTLMLogin	Opnum: 6

The object exporting this interface also implements IWbemLoginClientID and IWbemLogin helper interface. **IRemUnknown** and **IRemUnknown2** interfaces, as specified in [MS-DCOM], MUST be used to manage the interfaces exposed by the object. The object MUST be uniquely identified with CLSID {8BC3F05E-D86B-11D0-A075-00C04FB68820}.

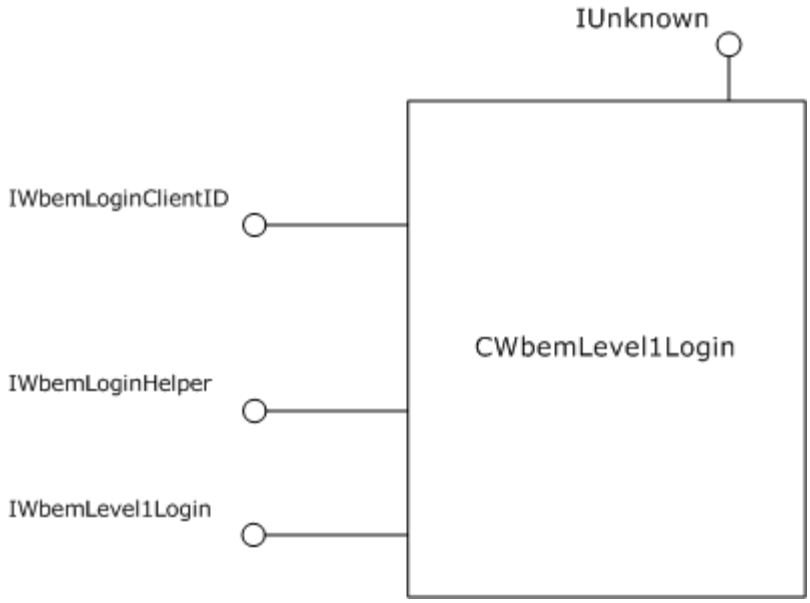


Figure 4: The IWbemLevel1Login interface

3.1.5.1.1 IWbemLevel1Login::EstablishPosition (Opnum 3)

The **IWbemLevel1Login::EstablishPosition** method does not perform any action.<7>

```
HRESULT EstablishPosition(
  [in, unique, string] wchar_t* wszLocaleList,
  [in] DWORD dwNumLocales,
  [out] DWORD* reserved
);
```

Return value/code	Description
0x00 WBEM_S_NO_ERROR	All versions of Windows beginning with Windows Vista MUST return this value.
0x80004001 E_NOTIMPL	All versions of Windows prior to Windows Vista MUST return this value.

3.1.5.1.2 IWbemLevel1Login::RequestChallenge (Opnum 4)

The **IWbemLevel1Login::RequestChallenge** method MUST return WBEM_E_NOT_SUPPORTED.<8>

This method does not perform any action.

```
HRESULT RequestChallenge(
  [in, unique, string] wchar_t* wszNetworkResource,
  [in, unique, string] wchar_t* wszUser,
  [out, size_is(16), length_is(16)]
    unsigned char* Nonce
```

```
);
```

Return value/code	Description
0x8004100c WBEM_E_NOT_SUPPORTED	All versions of Windows MUST return this value

3.1.5.1.3 IWbemLevel1Login::WBEMLogin (Opnum 5)

The **IWbemLevel1Login::WBEMLogin** method MUST return E_NOTIMPL.<9>

This method does not perform any action.

```
HRESULT WBEMLogin(
  [in, unique, string] wchar_t* wszPreferredLocale,
  [in, size_is(16), length_is(16), unique]
    unsigned char* AccessToken,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [out] IWbemServices** ppNamespace
);
```

Return value/code	Description
0x80004001 E_NOTIMPL	All versions of Windows MUST return this value.

3.1.5.1.4 IWbemLevel1Login::NTLMLogin (Opnum 6)

The **IWbemLevel1Login::NTLMLogin** method MUST connect a user to the management services interface in a specified namespace.

```
HRESULT NTLMLogin(
  [in, unique, string] LPWSTR wszNetworkResource,
  [in, unique, string] LPWSTR wszPreferredLocale,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [out] IWbemServices** ppNamespace
);
```

wszNetworkResource: The string MUST represent the namespace on the server to whom the returned IWbemServices is associated. This parameter MUST NOT be NULL and MUST match the namespace syntax as specified in section 2.2.1.2.

wszPreferredLocale: MUST be a pointer to a string that MUST specify the locale. The string format MUST be "MS_xxx", where "xxx" is a string representation of LCID in BASE16, which MUST identify the locale, as specified in [MS-LCID]. Any subsequent calls requesting **CIM Localizable Information** (WBEM_FLAG_USE_AMENDED_QUALIFIERS) should return the localized information if available in the LCID language.

lFlags: Flags that affect the behavior of the **NTLMLogin** method. The connection flags to wszNetworkResource MUST be interpreted as follows:

Value	Meaning
WBEM_FLAG_CONNECT_REPOSITORY_ONLY 0x00000040	If this bit is set, the server will not consult dynamic class providers to provide classes. If this bit is not set, the server will consult dynamic class providers to provide classes.

Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an IWbemContext interface, which MUST contain additional information the client wants to pass to the server. If pCtx is NULL, the parameter MUST be ignored.

ppNamespace: If the call succeeds, ppNamespace MUST return a pointer to an IWbemServices interface pointer. This parameter MUST be set to NULL when an error occurs.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (as specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemLevel1Login::NTLMLogin** method MUST be called on the interface obtained as result of a Distributed Component Object Model (DCOM) Remote Protocol (as specified in [MS-DCOM]) activation specified in section 3.1.3.

The security principal making the call MUST have WBEM_REMOTE_ENABLE and WBEM_ENABLE accesses to the namespace.

In response to **IWbemLevel1Login::NTLMLogin** method, the server MUST return an **IWbemServices** interface corresponding to wszNetworkResource parameter. The *wszPreferredLocale* parameter is stored on the Distributed Component Object Model (DCOM) Remote Protocol (as specified in [MS-DCOM]) **IWbemServices**.

All subsequent **IWbemServices** method invocations requesting localized information MUST return the information in the language specified in wszPreferredLocale. When the preferred locale is NULL, the server MUST use the default server language.

The successful method execution MUST fill the new **IWbemServices** interface pointer in ppWorkingNamespace parameter and MUST return WBEM_S_NO_ERROR.

The failed method execution MUST set output parameters to NULL and MUST return an error in the format as specified in section 2.2.1.3.

3.1.5.2 IWbemServices Interface

The **IWbemServices** interface exposes methods that MUST provide management services to client processes. The interface MUST be implemented by the server. The implementation MUST implement all methods and return errors if the semantics of the operation cannot be completed. **IWbemServices** defines the execution scope for all methods implemented on the interface. The initial scope MUST be established by the **IWbemLevel1Login::NTLMLogin** call, which returns the interface pointer.

Methods in RPC Opnum Order

Method	Description
OpenNamespace	Provides the client with an **IWbemServices** interface pointer scoped to the requested namespace. Opnum: 3
CancelAsyncCall	Cancels a currently pending asynchronous method call identified by the IWbemObjectSink pointer passed to the initial asynchronous method. Opnum: 4
QueryObjectSink	Obtains a notification handler that allows the client to send events directly to the server. Opnum: 5
GetObject	Retrieves a CIM class or a CIM instance. Opnum: 6
GetObjectAsync	Asynchronous version of the **IWbemServices::GetObject** method. Opnum: 7
PutClass	Creates a new class or updates an existing class into the namespace associated with the current **IWbemServices** interface. Opnum: 8
PutClassAsync	Asynchronous version of the **IWbemServices::PutClass** method. Opnum: 9
DeleteClass	Deletes a specified class from the namespace associated with the current **IWbemServices** interface. Opnum: 10
DeleteClassAsync	Asynchronous version of the **IWbemServices::DeleteClass** method. Opnum: 11
CreateClassEnum	Creates a class enumeration. Opnum: 12
CreateClassEnumAsync	Asynchronous version of the IWbemServices::CreateClassEnum method. Opnum: 13
PutInstance	Creates or updates an instance of an existing class. Opnum: 14
PutInstanceAsync	Asynchronous version of the PutInstance method. Opnum: 15
DeleteInstance	Deletes an instance of an existing class from the namespace that is pointed to by the **IWbemServices** interface object that is used to call the method. Opnum: 16
DeleteInstanceAsync	Asynchronous version of the **IWbemServices::DeleteInstance** method.

Method	Description
	Opnum: 17
CreateInstanceEnum	Creates an instance enumeration of all class instances that satisfy the selection criteria. Opnum: 18
CreateInstanceEnumAsync	Asynchronous version of the IWbemServices::CreateInstanceEnum method. Opnum: 19
ExecQuery	Returns an enumerable collection of **IWbemClassObject** interface objects based on a query. Opnum: 20
ExecQueryAsync	Asynchronous version of the **IWbemServices::ExecQuery** method. Opnum: 21
ExecNotificationQuery	Executes a query to receive events when called by a client to request subscription to the events. Opnum: 22
ExecNotificationQueryAsync	Asynchronous version of the **IWbemServices::ExecNotificationQuery** method. Opnum: 23
ExecMethod	Executes a **CIM method** implemented by a CIM class or a CIM instance retrieved from the **IWbemServices** interface. Opnum: 24
ExecMethodAsync	Asynchronous version of the IWbemServices::ExecMethod method. Opnum: 25

IWbemServices MUST be a DCOM Remote Protocol interface. The interface MUST be uniquely identified by UUID 9556dc99-828c-11cf-a37e-00aa003240c7. The object exporting this interface also implements the **IWbemRefreshingServices** interface, as shown in the following diagram.

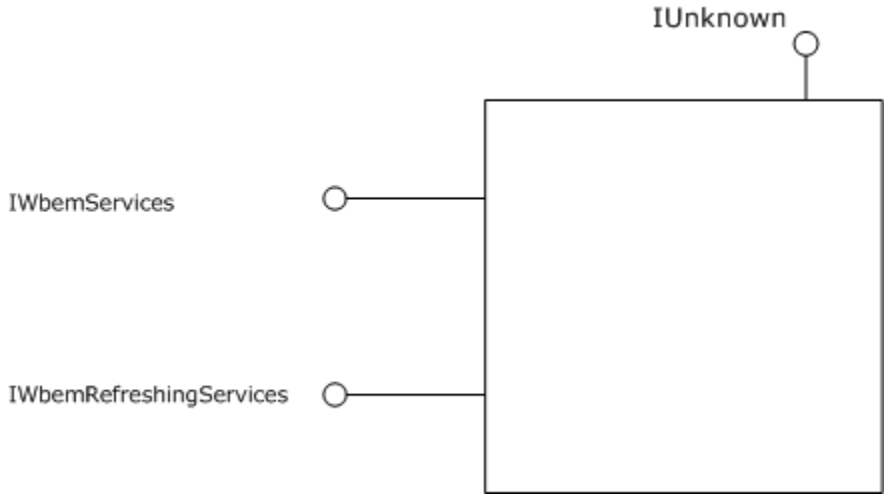


Figure 5: The IWbemServices interface

3.1.5.2.1 IWbemServices::OpenNamespace (Opnum 3)

The **IWbemServices::OpenNamespace** method provides the client with an **IWbemServices** interface pointer scoped to the requested namespace. The specified namespace MUST be a child namespace of the current namespace through which this method is called.

```
HRESULT OpenNameSpace(
  [in] const BSTR strNamespace,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [in, out, unique] IWbemServices** ppWorkingNamespace,
  [in, out, unique] IWbemCallResult** ppResult
);
```

strNamespace: : It MUST be the CIM path to the target namespace. This namespace MUST be relative to the current namespace associated with the **IWbemServices** interface pointer used to make the **OpenNamespace** method call. This parameter MUST NOT be NULL.

lFlags: Flags that affect the behavior of the **OpenNamespace** method. The flag's behavior MUST be interpreted as specified as follows:

- If this bit is not set, the server MUST make the method call synchronous.
- If this bit is set, the server MUST make the method call semisynchronously.

Name	Value
WBEM_FLAG_RETURN_IMMEDIATELY	0x00000010

Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: This parameter MUST be NULL.

ppWorkingNamespace: This parameter MUST NOT be NULL on input when WBEM_FLAG_RETURN_IMMEDIATELY is not set. In such a case, it MUST receive a pointer to an **IWbemServices** interface pointer to the requested namespace.

The output parameter MUST to the state of the **lFlags** parameter (whether WBEM_FLAG_RETURN_IMMEDIATELY is set) as listed in the following table:

Flag state	Success operation	Failure operation
WBEM_FLAG_RETURN_IMMEDIATELY is not set.	MUST be set to the requested IWbemServices interface.	MUST be set to NULL if the input parameter is not-NULL.
WBEM_FLAG_RETURN_IMMEDIATELY is set.	MUST be set to NULL if the input parameter is not-NULL.	MUST be set to NULL if the input parameter is not-NULL.

ppResult: The output parameter MUST be filled according to the state of the **lFlags** parameter (whether WBEM_FLAG_RETURN_IMMEDIATELY is set) as listed in the following table:

Flag state	Success operation	Failure operation
WBEM_FLAG_RETURN_IMMEDIATELY is not set.	MUST be set to NULL if the input parameter is not-NULL.	MUST be set to NULL if the input parameter is not-NULL.
WBEM_FLAG_RETURN_IMMEDIATELY is set.	MUST be set to the requested IWbemCellResult interface.	MUST be set to NULL if the input parameter is not-NULL.

This parameter MUST NOT be NULL on input when WBEM_FLAG_RETURN_IMMEDIATELY equals 1. In such a case, it receives a pointer to an IWbemCallResult interface pointer.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (as specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

In response to the **IWbemServices::OpenNamespace** method, the server MUST evaluate whether the *strNamespace* parameter (specified earlier in this section) is a child of the namespace associated with the current interface pointer and, on success, creates another **IWbemServices** interface pointer associated with this namespace.

If the method returns a success code, the method MUST fill one of the two output parameters, as indicated by the use of lFlags parameter, specified above.

The successful synchronous method execution MUST fill the new **IWbemServices** interface pointer in ppWorkingNamespace parameter and MUST return WBEM_S_NO_ERROR.

The successful semisynchronous method execution MUST return immediately, it MUST fill the *ppResultParameter* with a new **IWbemCallResult** interfaces pointer and MUST return WBEM_S_NO_ERROR. The final result of the operation MUST be retrieved using the sequences specified in section 3.1.5.4.3 and section 3.1.5.4.4.

The failed method execution MUST set the output parameters to NULL and MUST return an error in the format specified in section 2.2.1.11.

3.1.5.2.2 IWbemServices::CancelAsyncCall (Opnum 4)

The **IWbemServices::CancelAsyncCall** method SHOULD be used to cancel a currently pending asynchronous method call identified by the IWbemObjectSink pointer passed to the initial asynchronous method.

```
HRESULT CancelAsyncCall(
  [in] IWbemObjectSink* pSink
);
```

pSink: MUST be a pointer to the IWbemObjectSink interface object that was passed to the asynchronous method that the client wants to cancel. This parameter MUST NOT be NULL.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (as specified in section 2.2.1.11) to indicate the successful completion of the method.

Return value/code	Description
0x00 WBEM_S_NO_ERROR	Indicates a successful completion to the method call.

In response to the **IWbemServices::CancelAsyncCall** method, the server MUST identify and cancel all pending asynchronous operations initiated by an asynchronous method execution, such as **IWbemServices::GetObjectAsync**, which used the pSink interface pointer parameter as their response handler. The server MUST return an error if the interface pointer is NULL, and it MUST return an error if the *pSink* parameter is not associated with any pending operation.

The server MUST NOT wait for any response from the client to complete the cancellation to prevent protocol performance degradation.

The successful method execution MUST return WBEM_S_NO_ERROR.

The failed method execution MUST return an error in the format specified in section 2.2.1.11.

3.1.5.2.3 IWbemServices::QueryObjectSink (Opnum 5)

The **QueryObjectSink** method SHOULD be called to obtain a notification handler that allows the client to send events directly to the server. The client MUST only send extrinsic events to the server.

```
HRESULT QueryObjectSink(
  [in] long lFlags,
  [out] IWbemObjectSink** ppResponseHandler
);
```

lFlags: This parameter is not used, and its value MUST be 0x0.

ppResponseHandler: MUST be a pointer to an **QueryObjectSink** interface pointer to the notification handler that allows the client to send events directly to the server. This parameter MUST be set to NULL on error.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_REMOTE_ENABLE and WBEM_FULL_WRITE accesses to the namespace.

In response to the **IWbemServices::QueryObjectSink** method, the server MUST return a new **IWbemObjectSink** interface pointer. The client MUST use the returned interface to send Windows Management Instrumentation Remote Protocol events to the server. The server MUST return an error if the *ppHandler* is NULL, or if it is unable to create the requested interface pointer.

The successful method execution MUST fill *ppHandler* parameter and MUST return WBEM_S_NO_ERROR.

The failed method execution MUST set the output parameters to NULL and MUST return an error in the format as specified in section 2.2.1.11.

3.1.5.2.4 IWbemServices::GetObject (Opnum 6)

The **IWbemServices::GetObject** method retrieves a CIM class or a CIM instance. This method MUST retrieve CIM objects from the namespace associated with the current **IWbemServices** interface.

```
HRESULT GetObject(
  [in] const BSTR strObjectPath,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [out, in, unique] IWbemClassObject** ppObject,
  [out, in, unique] IWbemCallResult** ppCallResult
);
```

strObjectPath: MUST be the CIM path of the CIM object to be retrieved. If the parameter is NULL, the server MUST return an **empty CIM Object**.

lFlags: Flags that affect the behavior of the **IWbemServices::GetObject** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_USE_AMENDED_QUALIFIERS 0x00020000	If this bit is not set, the server SHOULD return no CIM Localizable Information. If this bit is set, the server SHOULD return CIM Localizable Information.
WBEM_FLAG_RETURN_IMMEDIATELY 0x00000010	If this bit is not set, the server MUST make the method call synchronously. If this bit is set, the server MUST make the method call semisynchronously.
WBEM_FLAG_DIRECT_READ 0x00000200	When this bit is set, the server MUST disregard any derived class when searching the result. When this bit is not set, the server MUST consider the entire class hierarchy when returning the result.

The flags MAY be combined in any form, provided that all restrictions provided in the flags description are respected. Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an **IWbemCoIDL_ntext** interface, which MUST contain additional information the client wants to pass for processing to the implementer of the CIM object referred by strObjectPath. If the parameter is set to NULL, the server MUST ignore it.

ppObject: If the parameter is set to NULL, the server MUST ignore it. When the parameter is not NULL, it MUST return an IWbemClassObject interface pointer containing the requested CIM object. In this case, the output parameter MUST be filled according to the state of the **lFlags** parameter (whether WBEM_FLAG_RETURN_IMMEDIATELY is set) as listed in the following table:

Flag state	Success operation	Failure operation
WBEM_FLAG_RETURN_IMMEDIATELY is	MUST contain an	MUST be set to NULL if

Flag state	Success operation	Failure operation
not set.	IWbemClassObject interface pointer.	the input parameter is not-NULL.
WBEM_FLAG_RETURN_IMMEDIATELY is set.	MUST be set to NULL if the input parameter is not-NULL.	MUST be set to NULL if the input parameter is not-NULL.

ppCallResult: This parameter is optional. If the parameter is set to NULL, the server MUST ignore it. However, when the parameter is not NULL, it MUST return an **IWbemCallResult** interface pointer, which SHOULD be polled to obtain the operation call result. In this case, the output parameter MUST be filled according to the state of the **lFlags** parameter (whether WBEM_FLAG_RETURN_IMMEDIATELY is set) as listed in the following table:

Flag state	Success operation	Failure operation
WBEM_FLAG_RETURN_IMMEDIATELY is not set	MUST be set to NULL if the input parameter is not-NULL.	MUST be set to NULL if the input parameter is not-NULL.
WBEM_FLAG_RETURN_IMMEDIATELY is set.	MUST contain the IWbemCallResult interface pointer.	MUST be set to NULL if the input parameter is not-NULL.

Return Values: This method MUST return an HRESULT, which MUST indicate the status of the method call. HRESULT MUST have the type and values as specified in section 2.2.1.11. The server MUST return WBEM_S_NO_ERROR (as specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_REMOTE_ENABLE and WBEM_ENABLE accesses to the namespace.

In response to **IWbemServices::GetObject** method, the server MUST evaluate *strObjectPath* parameter and it MUST return the requested CIM object. The method MUST fail if the CIM object referred by *strObjectPath* does not exist, if the method parameters are not valid, as specified above, or if the server is unable to execute the method.

The successful synchronous method execution MUST provide the retrieved **IEnumWbemClassObject** interface pointer in ppObject parameter and MUST return WBEM_S_NO_ERROR.

The successful semisynchronous method execution MUST complete immediately, it MUST fill the *ppCallResult* with a new **IWbemCallResult** interfaces pointer and MUST return WBEM_S_NO_ERROR. The final result of the operation MUST be retrieved by using the sequences as specified in section 3.1.5.4.1.

The failed method execution MUST set the output parameters to NULL and MUST return an error in the format specified in section 2.2.1.11.

3.1.5.2.5 IWbemServices::GetObjectAsync (Opnum 7)

The **GetObjectAsync** method is the asynchronous version of **IWbemServices::GetObject** method.

```
HRESULT GetObjectAsync(
  [in] const BSTR strObjectPath,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [in] IWbemObjectSink* pResponseHandler
);
```

strObjectPath: MUST be the CIM path of the CIM object to be retrieved. If set to NULL, the server MUST return an empty CIM Object.

lFlags: A set of flags that affect the behavior of the **GetObjectAsync** method. The flags behavior MUST be interpreted as specified in the following table.

The flags MAY be combined in any form, provided that all restrictions provided in the flags description are respected. Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an **IWbemContext** interface, which MUST contain additional information the client wants to provide to the server about the CIM object referred by strObjectPath. If pCtx is NULL, the parameter MUST be ignored.

Value	Meaning
WBEM_FLAG_USE_AMENDED_QUALIFIERS 0x00020000	If this bit is not set, the server SHOULD return no CIM Localizable Information. If this bit is set, the server SHOULD return CIM Localizable Information.
WBEM_FLAG_SEND_STATUS 0x00000080	If this bit is not set, the server MUST make one final **IWbemObjectSink::SetStatus** call on the interface pointer provided in pResponseHandler parameter. If this bit is set, the server MAY make intermediate **IWbemObjectSink::SetStatus** calls on the interfaces pointer prior to call completion.
WBEM_FLAG_DIRECT_READ 0x00000200	When this bit is not set, the implementor MUST consider the entire class hierarchy when returning the result. When this bit is set, the server MUST disregard any derived class when searching the result.

pResponseHandler: MUST be a pointer to **IWbemObjectSink Interface** implemented by the caller, where enumeration results must be delivered. The parameter MUST NOT be NULL. In error cases, indicated by the return value, the supplied **IWbemObjectSink Interface** interface pointer MUST NOT be used and the client MAY release it when the call is complete. If WBEM_S_NO_ERROR is returned, then the user's **IWbemObjectSink Interface** interface pointer MUST be called to indicate the result of the **GetObjectAsync** operation.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_REMOTE_ENABLE and WBEM_ENABLE accesses to the namespace.

In response to the **GetObjectAsync** method, the server MUST evaluate *strObjectPath* parameter, and it MUST return the requested CIM object. The method MUST fail if the CIM object referred by *strObjectPath* does not exist, if the method parameters are not valid as specified in this section, or if the server is unable to execute the method.

The successful asynchronous method execution MUST return immediately and MUST return WBEM_S_NO_ERROR. The server MUST continue method execution after the WBEM_S_NO_ERROR return and MUST complete the operation using the message sequence specified in section 3.1.1.3.

3.1.5.2.6 IWbemServices::PutClass (Opnum 8)

The **IWbemServices::PutClass** method SHOULD be called to create a new class or updates an existing class into the namespace associated with the current **IWbemServices** interface. The user MUST NOT create classes with names that begin or end with an underscore, because they are reserved for system classes.

```
HRESULT PutClass(
  [in] IWbemClassObject* pObject,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [out, in, unique] IWbemCallResult** ppCallResult
);
```

pObject: MUST be a pointer to an IWbemClassObject interface pointer which MUST contain the class information to create a new class or update an existing class. This parameter MUST NOT be NULL.

lFlags: Flags that affect the behavior of the **PutClass** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_USE_AMENDED_QUALIFIERS 0x00020000	If this bit is not set, the server SHOULD not return to CIM Localizable Information. If this bit is set, the server SHOULD return CIM Localizable Information.
WBEM_FLAG_RETURN_IMMEDIATELY 0x00000010	If this bit is not set, the server MUST make the method call synchronously. If this bit is set, the server MUST make the method call semisynchronously.
WBEM_FLAG_UPDATE_ONLY 0x00000001	The server MUST update a CIM class pObject if the CIM class is present. This flag is mutually exclusive with

Value	Meaning
	WBEM_FLAG_CREATE_ONLY. If none of these flags are set, the server MUST create or update a CIM class pObject.
WBEM_FLAG_CREATE_ONLY 0x00000002	The server MUST create a CIM class pObject if the CIM class is not already present.
WBEM_FLAG_UPDATE_FORCE_MODE 0x00000040	The server MUST forcefully update the class even when conflicting child classes exist.
WBEM_FLAG_UPDATE_SAFE_MODE 0x00000020	The server MUST update the class even if there are child classes, as long as the change does not cause any conflicts with child classes. If the class has instances, the update MUST fail. This flag is mutually exclusive with WBEM_FLAG_UPDATE_FORCE_MODE. If none of these flags are set, the server MUST update the class if there is no derived class, no instance for that class, or the class is unchanged.

The flags MAY be combined in any form, provided that all restrictions provided in the flags description are respected. Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an IWbemContext interface, which MUST contain additional information the client wants to provide to the server about the CIM class referred by pObject. If pCtx is NULL, the parameter MUST be ignored.

ppCallResult: The output parameter MUST be filled according to the state of the **lFlags** parameter (whether WBEM_FLAG_RETURN_IMMEDIATELY is set) as listed in the following table:

Flag state	Success operation	Failure operation
WBEM_FLAG_RETURN_IMMEDIATELY is not set	MUST be set to NULL if the input parameter is not-NULL.	MUST be set to NULL if the input parameter is not-NULL.
WBEM_FLAG_RETURN_IMMEDIATELY is set	MUST contain the IWbemCallResult interface pointer.	MUST be set to NULL if the input parameter is not-NULL.

This parameter is optional. If the parameter is set to NULL, the server MUST ignore it. However, when the parameter is not NULL, the server MUST return an **IWbemCallResult** interface pointer, which SHOULD be polled to obtain the operation call result.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_REMOTE_ENABLE and WBEM_FULL_WRITE accesses to the namespace.

In response to the **IWbemServices::PutClass** method, the server MUST evaluate the *pObject* parameter and MUST add or update this class into the current namespace. The method MUST fail if pObject represents a read-only class, if the method parameters or their combinations are not valid as specified in this section, or if the server is unable to execute the method.

The successful synchronous method execution MUST return WBEM_S_NO_ERROR.

The successful semisynchronous method execution MUST return immediately, it MUST fill the *ppCallResult* with a new **IWbemCallResult** interfaces pointer, and it MUST return WBEM_S_NO_ERROR. The final result of the operation MUST be retrieved by using the sequences specified in section 3.1.5.4.3 and section 3.1.5.4.4.

The failed method execution MUST set output parameters on NULL and MUST return an error in the format specified in section 2.2.1.11.

3.1.5.2.7 IWbemServices::PutClassAsync (Opnum 9)

The **IWbemServices::PutClassAsync** method is the asynchronous version of the **IWbemServices::PutClass** method. The **PutClassAsync** method SHOULD be called to create a new class or update an existing class. The user MUST NOT create classes with names that begin or end with an underscore because it is reserved for system classes.

```
HRESULT PutClassAsync(
  [in] IWbemClassObject* pObject,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [in] IWbemObjectSink* pResponseHandler
);
```

pObject: MUST be a pointer to an IWbemClassObject interface pointer, which MUST contain the class information to create a new class or update an existing class. The class specified by the parameter MUST have been correctly initialized with all of the required property values. This parameter MUST NOT be NULL.

lFlags: Flags that affect the behavior of the **PutClassAsync** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_USE_AMENDED_QUALIFIERS 0x00020000	If this bit is not set, the server SHOULD to return no CIM Localizable Information. If this bit is set, the server SHOULD return CIM Localizable Information.
WBEM_FLAG_UPDATE_ONLY 0x00000001	The server MUST update a CIM class pObject if the CIM class is present. This flag is mutually exclusive with WBEM_FLAG_CREATE_ONLY. If none of these flags are set, the server MUST create or update a CIM class pObject.
WBEM_FLAG_CREATE_ONLY 0x00000002	The server MUST create a CIM class pObject if the CIM class is not already present.
WBEM_FLAG_UPDATE_FORCE_MODE	The server MUST forcefully update the class even

Value	Meaning
0x00000040	when conflicting child classes exist.
WBEM_FLAG_UPDATE_SAFE_MODE 0x00000020	The server MUST update the class even if there are child classes, as long as the change does not cause any conflicts with child classes. If the class has instances, the update MUST fail. This flag is mutually exclusive with WBEM_FLAG_UPDATE_FORCE_MODE. If none of these flags are set, the server MUST update the class if there is no derived class, no instance for that class or the class is unchanged.
WBEM_FLAG_SEND_STATUS 0x00000080	If this bit is not set the server MUST make just one final IWbemObjectSink::SetStatus call on the interface pointer provided in pResponseHandler parameter. If this bit is set, the server MAY make intermediate IWbemObjectSink::SetStatus calls on the interfaces pointer prior to call completion.

The flags MAY be combined in any form, provided that all restrictions provided in the flags description are respected. Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an IWbemContext interface, which MUST contain additional information the client wants to pass to the server. If pCtx is NULL, the parameter MUST be ignored.

pResponseHandler: MUST be a pointer to an **IWbemObjectSink** interface object implemented by the client of this method. The parameter MUST NOT be NULL. In error cases, indicated by the HRESULT return value, the supplied **IWbemObjectSink** interface pointer MUST NOT be used and the client MAY release it when the call is complete. If the return value is equal to WBEM_S_NO_ERROR (as specified in section 2.2.1.11), then the user's **IWbemObjectSink** interface pointer MUST be used to indicate the result of the **PutClassAsync** operation, as specified earlier in this section.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_REMOTE_ENABLE and WBEM_FULL_WRITE accesses to the namespace.

In response to **IWbemServices::PutClassAsync** method, the server MUST evaluate the *pObject* parameter (specified above) and it MUST add or update this class into the current namespace. The method MUST fail if *pObject* represents a read-only class, if the method parameters or their combinations are not valid as specified above, or if the server is unable to execute the method.

The successful asynchronous method execution MUST complete immediately and MUST return WBEM_S_NO_ERROR. The server MUST continue method execution after the WBEM_S_NO_ERROR return and MUST complete the operation using the message sequence specified in section 3.1.1.3.

3.1.5.2.8 IWbemServices::DeleteClass (Opnum 10)

The **IWbemServices::DeleteClass** method MUST delete a specified class from the namespace associated with the current **IWbemServices** interface.

```
HRESULT DeleteClass(
  [in] const BSTR strClass,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [in, out, unique] IWbemCallResult** ppCallResult
);
```

strClass: It MUST be the name of the class to delete. This parameter MUST NOT be NULL.

lFlags: Flags that affect the behavior of the **DeleteClass** method.

The flags behavior MUST be interpreted as following:

Value	Meaning
WBEM_FLAG_RETURN_IMMEDIATELY 0x00000010	If this bit is set, the server MUST make the method call semisynchronously. If this bit is not set, the server MUST make the method call synchronously.

Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an **IWbemContext** interface, which MUST contain additional information the client wants to pass to the server. If pCtx is NULL, the parameter MUST be ignored.

ppCallResult: This parameter is optional. If the parameter is set to NULL, the server MUST ignore it. However, when the parameter is not NULL, the server MUST return an **IWbemCallResult** interface pointer, which SHOULD be polled to obtain the operation call result. The output parameter MUST be filled according to the state of the **lFlags** parameter (whether WBEM_FLAG_RETURN_IMMEDIATELY is set) as listed in the following table:

Flag state	Success operation	Failure operation
WBEM_FLAG_RETURN_IMMEDIATELY is not set.	MUST be set to NULL if the input parameter is not-NULL.	MUST be set to NULL if the input parameter is not-NULL.
WBEM_FLAG_RETURN_IMMEDIATELY is set.	MUST contain the IWbemCallResult interface pointer.	MUST be set to NULL if the input parameter is not-NULL.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_REMOTE_ENABLE and WBEM_FULL_WRITE accesses to the namespace.

In response to the **IWbemServices::DeleteClass** method, the server MUST evaluate *strClass* parameter (specified in this section) and MUST delete the *strClass* from the current namespace. The method MUST fail if *strClass* does not exist, if the method parameters or their combinations are not valid as specified in this section, or if the server is unable to execute the method.

The successful synchronous method execution MUST return WBEM_S_NO_ERROR.

The successful semisynchronous method execution MUST return immediately, it MUST fill the *ppCallResult* with a new **IWbemCallResult** interfaces pointer and MUST return WBEM_S_NO_ERROR. The final result of the operation MUST be retrieved using the sequences specified in section 3.1.1.2 below.

The failed method execution MUST set output parameters on NULL and MUST return an error in the format specified in section 2.2.1.11.

3.1.5.2.9 IWbemServices::DeleteClassAsync (Opnum 11)

The **IWbemServices::DeleteClassAsync** method is the asynchronous version of the **IWbemServices::DeleteClass** method. The **DeleteClassAsync** method MUST delete a specified class from the namespace.

```
HRESULT DeleteClassAsync(
  [in] const BSTR strClass,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [in] IWbemObjectSink* pResponseHandler
);
```

strClass: It MUST be the name of the class to delete. This parameter MUST NOT be NULL.

lFlags: Flags that affect the behavior of the **DeleteClassAsync** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_SEND_STATUS 0x00000080	If this bit is not set the server MUST make just one final **IWbemObjectSink::SetStatus** call on the interface pointer provided in pResponseHandler parameter. If this bit is set, the server MAY make intermediate **IWbemObjectSink::SetStatus** calls on the interfaces pointer prior to call completion.

Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an IWbemContext interface, which MUST contain additional information the client wants to pass to the server. If pCtx is NULL, the parameter MUST be ignored.

pResponseHandler: MUST be a pointer to an **IWbemObjectSink** interface object implemented by the client of this method. This parameter MUST NOT be NULL. In error cases, indicated by the HRESULT return value, the supplied IWbemObjectSink interface pointer MUST NOT be used and the client MAY release it when the call is complete. If the return value is equal to

WBEM_S_NO_ERROR (as specified in section 2.2.1.11), then the user's **IWbemObjectSink** interface pointer MUST be used to indicate the result of the **DeleteClassAsync** operation, as specified in section 3.1.1.3.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (as specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_REMOTE_ENABLE and WBEM_FULL_WRITE accesses to the namespace.

In response to the **IWbemServices::DeleteClassAsync** method, the server MUST evaluate *strClass* parameter (specified in this section) and MUST delete the strClass from the current namespace. The method MUST fail if *strClass* does not exist, if the method parameters or their combinations are not valid as specified above, or if the server is unable to execute the method.

The successful asynchronous method execution MUST return immediately and MUST return WBEM_S_NO_ERROR. The server MUST continue method execution after the WBEM_S_NO_ERROR return and MUST complete the operation using the message sequence specified in section 3.1.1.3.

3.1.5.2.10 IWbemServices::CreateClassEnum (Opnum 12)

The **IWbemServices::CreateClassEnum** method SHOULD be called by the client if the client desires to request a class enumeration. In response, the server MUST return all classes satisfying the selection criteria from the namespace associated with the current **IWbemServices** interface.

```
HRESULT CreateClassEnum(
  [in] const BSTR strSuperClass,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [out] IEnumWbemClassObject** ppEnum
);
```

strSuperClass: MUST specify a superclass name. Only classes that are subclasses of this class MUST be returned. If strSuperClass is NULL or a zero length string all classes in the namespace MUST be included in the result set. The results MUST be filtered using lFlags parameter. Classes without a base class MUST be considered to be derived from the NULL superclass.

lFlags: Flags affect the behavior of the **CreateClassEnum** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_USE_AMENDED_QUALIFIERS 0x00020000	If this bit is not set, the server SHOULD return no CIM Localizable Information. If this bit is set, the server SHOULD return CIM Localizable Information.
WBEM_FLAG_RETURN_IMMEDIATELY 0x00000010	If this bit is not set, the server MUST make the method call synchronously. If this bit is set, the server MUST make the method

Value	Meaning
	call semisynchronously.
WBEM_FLAG_SHALLOW 0x00000001	When this bit is not set the server MUST return all classes derived from the requested class and all its subclasses. When this bit is set, the server MUST return only classes directly derived from the requested class.
WBEM_FLAG_FORWARD_ONLY 0x00000020	When the bit is not set the server MUST return an enumerator with reset capability, When the bit is set, the server MUST return an enumerator without reset capability as specified in section 3.1.5.3.

The flags MAY be combined in any form, provided that all restrictions provided in the flags description are respected. Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an **IWbemContext** interface, which MUST contain additional information the client wants to pass to the server. If pCtx is NULL, the parameter MUST be ignored.

ppEnum: MUST be an out parameter, which MUST receive the pointer to the enumerator, implementing the **IEnumWbemClassObject** interface. This parameter MUST NOT be NULL.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (as specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_REMOTE_ENABLE and WBEM_ENABLE accesses to the namespace.

In response to the **IWbemServices::CreateClassEnum** method, the server MUST evaluate *strSuperClass* parameter (specified in this section) and MUST return all classes matching the input parameters from the current namespace. The method MUST fail if *strSuperClass* does not exist, if the method parameters or their combinations are not valid as specified above, or if the server is unable to execute the method.

The successful synchronous method execution MUST fill *ppEnum* parameter with a new **IEnumWbemClassObject** after all classes are collected and MUST return WBEM_S_NO_ERROR. The client MUST use **IEnumWbemClassObject** methods to obtain the real results of the operation using the sequences specified in section 3.1.5.3.2.

The successful semisynchronous method execution MUST return immediately, it MUST fill *ppEnum* parameter, and it MUST return WBEM_S_NO_ERROR. The final result of the operation MUST be retrieved using the sequences specified in section 3.1.5.3.2.

The failed method execution MUST set the value referenced by the output parameters to NULL and MUST return an error in the format specified in section 2.2.1.11.

3.1.5.2.11 IWbemServices::CreateClassEnumAsync (Opnum 13)

The **IWbemServices::CreateClassEnumAsync** method SHOULD be called by the client if the client desires to request an asynchronous class enumeration. In response, the server MUST return all classes that satisfy the selection criteria.

```
HRESULT CreateClassEnumAsync(
  [in] const BSTR strSuperClass,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [in] IWbemObjectSink* pResponseHandler
);
```

strSuperClass: Specifies a superclass name. Only classes that are subclasses of this class MUST be returned. If strSuperClass is NULL or a zero length string, all classes in the namespace MUST be considered in the result set. The results MUST be filtered using lFlags parameter. Classes without a base class are considered to be derived from the NULL superclass.

lFlags: Flags that affect the behavior of the **CreateClassEnum** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_USE_AMENDED_QUALIFIERS 0x00020000	If this bit is not set, the server SHOULD return no CIM Localizable Information. If this bit is set, the server SHOULD return CIM Localizable Information.
WBEM_FLAG_SEND_STATUS 0x00000080	If this bit is not set the server MUST make just one final IWbemObjectSink::SetStatus call on the interface pointer provided in pResponseHandler parameter. If this bit is set, the server MAY make intermediate IWbemObjectSink::SetStatus calls on the interfaces pointer prior to call completion.
WBEM_FLAG_SHALLOW 0x00000001	When this bit is not set the server MUST return all classes derived from the requested class and all its subclasses. When this bit is set, the server MUST return only classes directly derived from the requested class.

The flags MAY be combined in any form, provided that all restrictions provided in the flags description are respected. Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an **IWbemContext** interface, which MUST contain additional information the client wants to pass to the server. If pCtx is NULL, the parameter MUST be ignored.

pResponseHandler: MUST be a pointer to **IWbemObjectSink** implemented by the caller, where enumeration results must be delivered. The parameter MUST NOT be NULL. In error cases, indicated by the return value, the supplied **IWbemObjectSink** interface pointer MUST NOT be used. If WBEM_S_NO_ERROR is returned, then the user's **IWbemObjectSink** interface pointer MUST be called to indicate the results of **CreateClassEnumAsync** operation, as specified later in this section.

Return Values: This method MUST return an HRESULT, which MUST indicate the status of the method call. HRESULT MUST have the type and values as specified in section 2.2.1.11. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_REMOTE_ENABLE and WBEM_ENABLE accesses to the namespace.

In response to **IWbemServices::CreateClassEnumAsync** method, the server MUST evaluate *strSuperClass* parameter (specified in this section) and MUST return all classes matching the input parameters from the current namespace. The method MUST fail if *strSuperClass* does not exist, if the method parameters or their combinations are not valid as specified earlier in this section, or if the server is unable to execute the method.

The successful asynchronous method execution MUST complete immediately and MUST return WBEM_S_NO_ERROR. The server MUST continue method execution after the WBEM_S_NO_ERROR return and MUST complete the operation using the message sequence specified in section 3.1.1.3.

3.1.5.2.12 IWbemServices::PutInstance (Opnum 14)

The **IWbemServices::PutInstance** method SHOULD be called to create or update an instance of an existing class.

The **PutInstance** method opnum equals 14.

```
HRESULT PutInstance(
  [in] IWbemClassObject* pInst,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [in, out, unique] IWbemCallResult** ppCallResult
);
```

pInst: MUST be a pointer to an IWbemClassObject interface object, which MUST contain the class instance that the client wants to create or update. This parameter MUST NOT be NULL.

lFlags: : Flags that affect the behavior of the **PutInstance** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_USE_AMENDED_QUALIFIERS 0x00020000	If this bit is not set, the server SHOULD return no CIM Localizable Information. If this bit is set, the server SHOULD return CIM Localizable Information.
WBEM_FLAG_RETURN_IMMEDIATELY 0x00000010	If this bit is not set, the server MUST make the method call synchronously. If this bit is set, the server MUST make the method call semisynchronously.
WBEM_FLAG_UPDATE_ONLY 0x00000001	The server MUST update a CIM instance pObject if the CIM instance is present.

Value	Meaning
	This flag is mutually exclusive with WBEM_FLAG_CREATE_ONLY. If none of these flags are set, the server MUST create or update a CIM instance pObject.
WBEM_FLAG_CREATE_ONLY 0x00000002	The server MUST create a CIM instance pObject if the CIM instance is not already present.

The flags MAY be combined in any form, provided that all restrictions provided in the flags description are respected. Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: This parameter is optional. pCtx MUST be a pointer to an **IWbemContext** interface object. pCtx indicates whether the client is requesting a partial-instance update or full-instance update. A partial-instance update modifies a subset of the CIM instance's properties. In contrast, a full-instance update modifies all of the properties. If NULL, this parameter indicates that the client application is requesting a full-instance update. When pCtx is used to perform a partial-instance update, the IWbemContext interface object MUST be filled in with the properties specified in the following table.

Property name	Type	Description
__PUT_EXTENSIONS	VT_BOOL	If set to TRUE, it indicates that one or more of the other IWbemContext values have been specified. To perform a partial instance update, this property MUST be set to TRUE and properties below MUST be set as specified in their respective descriptions.
__PUT_EXT_STRICT_NULLS	VT_BOOL	If set to TRUE, the server MUST force the setting of properties to NULL. This parameter is optional.
__PUT_EXT_PROPERTIES	VT_ARRAY | VT_BSTR	Contains a CIM Property list to update. The server MUST ignore properties not listed. To perform a partial instance update, the list of properties MUST be specified.
__PUT_EXT_ATOMIC	VT_BOOL	On a success return code, all CIM Property updates MUST have been successful. On failure, the server MUST revert back the changes to the original state for all CIM Property updated. On failure, not a single change MUST remain. The operation is successful when all properties are updated.

ppCallResult: This parameter is optional. If the parameter is set to NULL, the server MUST ignore it. However, when the parameter is not NULL, the server MUST return an **IWbemCallResult** interface pointer, which SHOULD be polled to obtain the operation call result. The output parameter MUST be filled according to the state of the **lFlags** parameter (whether WBEM_FLAG_RETURN_IMMEDIATELY is set) as listed in the following table:

Flag state	Success operation	Failure operation
WBEM_FLAG_RETURN_IMMEDIATELY is not set.	MUST be set to NULL if the input parameter is not-	MUST be set to NULL if the input parameter is

Flag state	Success operation	Failure operation
	NULL.	not-NULL.
WBEM_FLAG_RETURN_IMMEDIATELY is set.	MUST contain the IWbemCallResult interface pointer.	MUST be set to NULL if the input parameter is not-NULL.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_REMOTE_ENABLE and WBEM_FULL_WRITE accesses to the namespace.

In response to the **IWbemServices::PutInstance** method, the server MUST evaluate the *pInst* parameter as (specified in this section). It MUST add or update this instance into the current namespace. The method MUST fail if the server does not allow creating new instances for the class of *pInst* or does not allow modification of the instance represented by *pInst*, if the method parameters or their combinations are not valid as specified in this section or if the server is unable to execute the method.

The successful synchronous method execution MUST return WBEM_S_NO_ERROR.

The successful semisynchronous method execution MUST complete immediately, it MUST fill the *ppCallResult* with a new **IWbemCallResult** interface pointer and MUST return WBEM_S_NO_ERROR. The final result of the operation MUST be retrieved using the sequences specified in section 3.1.5.4.4.

The failed method execution MUST set output parameters on NULL and MUST return an error in the format specified in section 2.2.1.11.

3.1.5.2.13 IWbemServices::PutInstanceAsync (Opnum 15)

The **IWbemServices::PutInstanceAsync** method is the asynchronous version of the **PutInstance** method. The **PutInstanceAsync** method SHOULD be called to create or update an instance of an existing class

```
HRESULT PutInstanceAsync(
  [in] IWbemClassObject* pInst,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [in] IWbemObjectSink* pResponseHandler
);
```

pInst: MUST be a pointer to an IWbemClassObject interface object, which MUST contain the class instance that the client wants to create or update. This parameter MUST NOT be NULL.

lFlags: Flags that affect the behavior of the **PutInstanceAsync** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_USE_AMENDED_QUALIFIERS 0x00020000	If this bit is not set, the server SHOULD return no CIM Localizable Information. If this bit is set, the server SHOULD return CIM Localizable Information.
WBEM_FLAG_UPDATE_ONLY 0x00000001	The server MUST update a CIM instance pObject if the CIM instance is present. This flag is mutually exclusive with WBEM_FLAG_CREATE_ONLY. If none of these flags are set, the server MUST create or update a CIM instance pObject.
WBEM_FLAG_CREATE_ONLY 0x00000002	The server MUST create a CIM object instance pObject if the CIM instance is not already present.
WBEM_FLAG_SEND_STATUS 0x00000080	If this bit is not set the server MUST make just one final **IWbemObjectSink::SetStatus** call on the interface pointer provided in pResponseHandler parameter. If this bit is set, the server MAY make intermediate **IWbemObjectSink::SetStatus** calls on the interfaces pointer prior to call completion.

The flags MAY be combined in any form, provided that all restrictions provided in the flags description are respected. Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: This parameter is optional. pCtx MUST be a pointer to an **IWbemContext** interface object specified in section 2.2.1.6. pCtx indicates whether the client is requesting a partial-instance update or full-instance update. A partial-instance update modifies a subset of the CIM instance's properties. In contrast, a full-instance update modifies all of the properties. If NULL, this parameter indicates that the client application is requesting a full-instance update. When pCtx is used to perform a partial-instance update, the IWbemContext Interface (section 2.2.1.14) MUST be filled in with the properties specified in the following table.

Property name	Type	Description
__PUT_EXTENSIONS	VT_BOOL	If set to TRUE, it indicates that one or more of the other IWbemContext values have been specified. To perform a partial instance update, this property MUST be set to TRUE and properties below MUST be set as specified in their respective descriptions.
__PUT_EXT_STRICT_NULLS	VT_BOOLL	If set to TRUE, the server MUST force the setting of properties to NULL. This parameter is optional.
__PUT_EXT_PROPERTIES	VT_ARRAY | VT_BSTR	Contains a CIM Property list to update. The server MUST ignore properties not listed. To perform a partial instance update, the list of properties MUST be specified.
__PUT_EXT_ATOMIC	VT_BOOL	On a success return code, all CIM Property updates MUST have been successful. On failure, the server MUST revert back the changes to the original state for all CIM Property updated. On

Property name	Type	Description
		failure, not a single change MUST remain. The operation is successful when all properties are updated.

pResponseHandler: MUST be a pointer to an **IWbemObjectSink** interface object implemented by the client of this method. This parameter MUST NOT be NULL. In error cases, indicated by the HRESULT return value, the supplied **IWbemObjectSink** interface pointer MUST NOT be used and the client MAY release it when the call is complete. If the return value is equal to WBEM_S_NO_ERROR (specified in section 2.2.1.11), then the user's **IWbemObjectSink** interface pointer MUST be used to indicate the result of the **PutInstanceAsync** operation as specified in section 3.1.5.5.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_REMOTE_ENABLE and WBEM_FULL_WRITE accesses to the namespace.

In response to an **IWbemServices::PutInstanceAsync** method, the server MUST evaluate the *pInst* parameter as specified in this section. It MUST add or update this instance into the current namespace. The method MUST fail if the server does not allow creating new instances for the class of pInst, or does not allow modification of the instance represented by *pInst*, if the method parameters, or their combinations are not valid as specified earlier in this section, or if the server is unable to execute the method.

The successful method execution MUST complete immediately. The server MUST use the provided pResponseHandler to report the status as specified in 3.1.1.3.

The failed method execution MUST return an error in the format specified in section 2.2.1.11.

3.1.5.2.14 IWbemServices::DeleteInstance (Opnum 16)

The **IWbemServices::DeleteInstance** method SHOULD be called to delete an instance of an existing class from the namespace that the **IWbemServices** interface object that is used to call the method points to.

```
HRESULT DeleteInstance(
  [in] const BSTR strObjectPath,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [in, out, unique] IWbemCallResult** ppCallResult
);
```

strObjectPath: MUST be the CIM path to the class instance that the client wants to delete. This parameter MUST NOT be NULL. The CIM path MUST contain the class name and the value of the key properties.

lFlags: Flags that affect the behavior of the **IWbemServices::DeleteInstance** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_RETURN_IMMEDIATELY 0x00000010	If this bit is not set, the server MUST make the method call synchronously. If this bit is set, the server MUST make the method call semisynchronously.

Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an **IWbemContext** interface, which MUST contain additional information the client wants to pass to the server. If pCtx is NULL, the parameter MUST be ignored.

ppCallResult: This parameter is optional. If the parameter is set to NULL, the server MUST ignore it. However, when the parameter is not NULL, the server MUST return an **IWbemCallResult** interface pointer, which SHOULD be polled to obtain the operation call result. The output parameter MUST be filled according to the state of the **lFlags** parameter (whether WBEM_FLAG_RETURN_IMMEDIATELY is set) as listed in the following table:

Flag state	Success operation	Failure operation
WBEM_FLAG_RETURN_IMMEDIATELY is not set.	MUST be set to NULL if the input parameter is not-NULL.	MUST be set to NULL if the input parameter is not-NULL.
WBEM_FLAG_RETURN_IMMEDIATELY is set.	MUST contain the IWbemCallResult interface pointer.	MUST be set to NULL if the input parameter is not-NULL.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_REMOTE_ENABLE and WBEM_FULL_WRITE accesses to the namespace.

In response to **IWbemServices::DeleteInstance** method, the server MUST evaluate *strObjectPath* parameter (as specified in this section) and MUST delete the instance identified by strObjectPath from the current namespace. The method MUST fail if *strObjectPath* does not exist, if the method parameters or their combinations are not valid as specified earlier in this section, or if the server is unable to execute the method.

The successful synchronous method execution MUST return WBEM_S_NO_ERROR.

The successful semisynchronous method execution MUST return immediately, it MUST fill the *ppCallResult* with a new **IWbemCallResult** interfaces pointer and MUST return WBEM_S_NO_ERROR. The final result of the operation MUST be retrieved using the sequences specified in section 3.1.1.2.

The failed method execution MUST set the output parameters to NULL and MUST return an error in the format specified in section 2.2.1.11.

3.1.5.2.15 IWbemServices::DeleteInstanceAsync (Opnum 17)

The **IWbemServices::DeleteInstanceAsync** method is the asynchronous version of the **IWbemServices::DeleteInstance** method. The **IWbemServices::DeleteInstanceAsync** method SHOULD be called to delete an instance of an existing class from the namespace that the **IWbemServices Interface** interface object that is used to call the method points to

```
HRESULT DeleteInstanceAsync(
  [in] const BSTR strObjectPath,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [in] IWbemObjectSink* pResponseHandler
);
```

strObjectPath: MUST be the CIM path to the class instance that the client wants to delete. This parameter MUST NOT be NULL. The CIM path MUST contain the class name and the value of the key properties.

lFlags: Flags that affect the behavior of the **IWbemServices::DeleteInstanceAsync** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_SEND_STATUS 0x00000080	If this bit is not set, the server MUST make just one final **IWbemObjectSink::SetStatus** call on the interface pointer provided in pResponseHandler parameter. If this bit is set, the server MAY make intermediate **IWbemObjectSink::SetStatus** calls on the interfaces pointer prior to call completion.

Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an IWbemContext interface, which MAY contain additional information the client wants to pass to the server. If pCtx is NULL, the parameter MUST be ignored.

pResponseHandler: MUST be a pointer to an **IWbemObjectSink** interface object implemented by the client of this method. This parameter MUST NOT be NULL. In error cases, indicated by the HRESULT return value, the supplied **IWbemObjectSink** interface pointer MUST NOT be used and the client MAY release it when the call is complete. If the return value is equal to WBEM_S_NO_ERROR (as specified in section 2.2.1.3), then the user's **IWbemObjectSink** interface pointer MUST be used to indicate the result of the **IWbemServices::DeleteInstanceAsync** operation, as specified in section **IWbemObjectSink Interface**.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_REMOTE_ENABLE and WBEM_FULL_WRITE accesses to the namespace.

In response to an **IWbemServices::DeleteInstanceAsync** method, the server MUST evaluate *strObjectPath* parameter (as specified in this section) and MUST delete the instance identified by *strObjectPath* from the current namespace. The method MUST fail if *strObjectPath* does not exist, if the method parameters or their combinations are not valid as specified in this section or if the server is unable to execute the method.

The successful method execution MUST complete immediately. The server MUST use the provided pResponseHandler to report the status as specified in 3.1.1.3.

The failed method execution MUST return an error in the format specified in section 2.2.1.11.

3.1.5.2.16 IWbemServices::CreateInstanceEnum (Opnum 18)

The **IWbemServices::CreateInstanceEnum** method SHOULD be called by the client if the client desires to request an instance enumeration. In response, the server MUST return all class instances that satisfy the selection criteria.

```
HRESULT CreateInstanceEnum(
  [in] const BSTR strFilter,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [out] IEnumWbemClassObject** ppEnum
);
```

strFilter: MUST contain the name of the CIM class for which the client wants instances of. This parameter MUST NOT be NULL.

lFlags: Flags that affect the behavior of the **CreateInstanceEnum** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_USE_AMENDED_QUALIFIERS 0x00020000	If this bit is not set, the server SHOULD return no CIM Localizable Information. If this bit is set, the server SHOULD return CIM Localizable Information.
WBEM_FLAG_RETURN_IMMEDIATELY 0x00000010	If this bit is not set, the server MUST make the method call synchronously. If this bit is set, the server MUST make the method call semisynchronously.
WBEM_FLAG_DIRECT_READ 0x00000200	When this bit is not set, the server MUST consider the entire class hierarchy when returning the result. When this bit is set, the server MUST disregard any derived class when searching the result.
WBEM_FLAG_SHALLOW 0x00000001	When this bit is not set the server MUST return all classes derived from the requested class and all its subclasses. When this bit is set, the server MUST return only classes directly derived from the requested class.
WBEM_FLAG_FORWARD_ONLY 0x00000020	When the bit is not set the server MUST return an enumerator with reset capability.

Value	Meaning
	When the bit is set, the server MUST return an enumerator without reset capability as specified in section 3.1.5.3.

The flags MAY be combined in any form, provided that all restrictions provided in the flags description are respected. Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an **IWbemContext** interface, which MAY contain additional information the client wants to pass to the server. If pCtx is NULL, the parameter MUST be ignored.

ppEnum: MUST be an out parameter which MUST receive the pointer to the enumerator that is used to enumerate through the returned class instances, which implements the **IEnumWbemClassObject** interface. This parameter MUST NOT be NULL.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return the following value(specified in section 2.2.1.11) below to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_ENABLE and WBEM_REMOTE_ENABLE accesses to the namespace.

In response to the **IWbemServices::CreateInstanceEnum** method, the server MUST evaluate the *strFilter* parameter (as specified in this section) and MUST return all instances for the specific class in the current namespace. The method MUST fail if *strFilter* does not exist, if the method parameters or their combinations are not valid as specified in this section, or if the server is unable to execute the method.

The successful synchronous method execution MUST fill *ppEnum* parameter with a new **IEnumWbemClassObject** after all instances are collected and MUST return WBEM_S_NO_ERROR. The client MUST use **IEnumWbemClassObject** methods to obtain the real results of the operation using the sequences specified in section 3.1.1.1.

The successful semisynchronous method execution MUST return immediately, it MUST fill *ppEnum* parameter, and it MUST return WBEM_S_NO_ERROR. The final result of the operation MUST be retrieved using the sequences specified in section 3.1.1.1.

The failed method execution MUST set the value referenced by the output parameters to NULL and MUST return an error in the format specified in section 2.2.1.11.

3.1.5.2.17 IWbemServices::CreateInstanceEnumAsync (Opnum 19)

The **IWbemServices::CreateInstanceEnumAsync** method SHOULD be called by the client if the client wants to request an asynchronous instance enumeration. In response, the server MUST return instances of all classes that satisfy the selection criteria.

```
HRESULT CreateInstanceEnumAsync(
  [in] const BSTR strSuperClass,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
```

```
  [in] IWbemObjectSink* pResponseHandler
);
```

strSuperClass: Specifies a superclass name. Only classes that are subclasses of this class MUST be returned. If strSuperClass is NULL or a zero length string, all classes in the namespace MUST be considered in the result set. The results MUST be filtered using lFlags parameter. Classes without a base class are considered to be derived from the NULL superclass.

lFlags: Flags that affect the behavior of the **IWbemServices::CreateInstanceEnumAsync** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_USE_AMENDED_QUALIFIERS 0x00020000	If this bit is not set, the server SHOULD return no CIM Localizable Information. If this bit is set, the server SHOULD return CIM Localizable Information.
WBEM_FLAG_SEND_STATUS 0x00000080	If this bit is not set the server MUST make just one final **IWbemObjectSink::SetStatus** call on the interface pointer provided in pResponseHandler parameter. If this bit is set, the server MAY make intermediate **IWbemObjectSink::SetStatus** calls on the interfaces pointer prior to call completion.
WBEM_FLAG_DIRECT_READ 0x00000200	When this bit is not set, the server MUST consider the entire class hierarchy when returning the result. When this bit is set, the server MUST disregard any derived class when searching the result.
WBEM_FLAG_SHALLOW 0x00000001	When this bit is not set the server MUST return all classes derived from the requested class and all its subclasses. When this bit is set, the server MUST return only classes directly derived from the requested class.

The flags MAY be combined in any form, given that all restrictions provided in the flags description are respected. Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an **IWbemContext** interface, which MUST contain additional information the client wants to pass to the server. If pCtx is NULL, the parameter MUST be ignored.

pResponseHandler: MUST be a pointer to **IWbemObjectSink** implemented by the caller, where enumeration results must be delivered. The parameter MUST NOT be NULL. In error cases, indicated by the return value, the supplied **IWbemObjectSink** interface pointer MUST NOT be used. If WBEM_S_NO_ERROR is returned, then the user's **IWbemObjectSink** interface pointer MUST be called to indicate the results of **CreateInstanceEnumAsync** operation, as specified later in this section.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (as specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_ENABLE and WBEM_REMOTE_ENABLE accesses to the namespace.

In response to **IWbemServices::CreateInstanceEnumAsync**, the server MUST evaluate the *strSuperClass* parameter (as specified in this section) and MUST return all instances for the given class in the current namespace. The method MUST fail if *strSuperClass* does not exist, if the method parameters or their combinations are not valid as specified earlier in this section, or if the server is unable to execute the method.

The successful asynchronous method execution MUST complete immediately and MUST return WBEM_S_NO_ERROR. The server MUST continue method execution after the WBEM_S_NO_ERROR return and MUST complete the operation using the message sequence specified in section 3.1.1.3.

The failed method execution MUST return an error in the format specified in section 2.2.1.11.

3.1.5.2.18 IWbemServices::ExecQuery (Opnum 20)

The **IWbemServices::ExecQuery** method returns an enumerable collection of IWbemClassObject interface objects based on a query.

```
HRESULT ExecQuery(
  [in] const BSTR strQueryLanguage,
  [in] const BSTR strQuery,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [out] IEnumWbemClassObject** ppEnum
);
```

strQueryLanguage: MUST contain one of the query languages supported by WMI. This parameter MUST NOT be NULL.<10>

strQuery: MUST contain the "WQL" query text as specified in [UNICODE] (UTF-16), as specified in section 2.2.1.1. This parameter MUST NOT be NULL.

lFlags: Flags that affect the behavior of the **IWbemServices::ExecQuery** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_USE_AMENDED_QUALIFIERS 0x00020000	If this bit is not set, the server SHOULD not return CIM Localizable Information. If this bit is set, the server SHOULD return CIM Localizable Information.
WBEM_FLAG_RETURN_IMMEDIATELY 0x00000010	If this bit is not set, the server MUST make the method call synchronously. If this bit is set, the server MUST make the method call semisynchronously.

Value	Meaning
WBEM_FLAG_DIRECT_READ 0x00000200	When this bit is not set, the server MUST consider the entire class hierarchy when returning the result. When this bit is set, the server MUST disregard any derived class when searching the result.
WBEM_FLAG_PROTOTYPE 0x00000002	When this bit is not set, the server SHOULD execute the query. When this bit is set, the server SHOULD NOT execute the query and instead return an object that looks like a typical result object.
WBEM_FLAG_FORWARD_ONLY 0x00000020	When the bit is not set the server MUST return an enumerator with reset capability, When the bit is set, the server MUST return an enumerator without reset capability, as specified in section 3.1.5.3.

The flags MAY be combined in any form, provided that all restrictions provided in the flags description are respected. Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an **IWbemContext** interface, which MUST contain additional information the client wants to pass to the server. If pCtx is NULL, the parameter MUST be ignored.

ppEnum: MUST be an out parameter, which MUST implement the **IEnumWbemClassObject** interface, which MUST receive the pointer to the enumerator that is used to enumerate through the CIM objects returned for the query result set. This parameter MUST NOT be NULL.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (as specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_ENABLE and WBEM_REMOTE_ENABLE accesses to the namespace.

In response to **IWbemServices::ExecQuery**, the server MUST evaluate the strQuery and strQueryLanguage parameters (as specified in this section) and MUST return all instances matching the provided query. The method MUST fail if the method parameters or their combinations are not valid as specified earlier in this section, or if the server is unable to execute the method.

The successful synchronous method execution MUST fill ppEnum parameter with a new **IEnumWbemClassObject** interface pointer after all instances are collected and MUST return WBEM_S_NO_ERROR. The client MUST use **IEnumWbemClassObject** methods to obtain the results of the operation using the sequences as specified in section 3.1.1.1.

The successful semisynchronous method execution MUST return immediately, it MUST fill ppEnum parameter with a new **IEnumWbemClassObject** interface pointer, and it MUST return WBEM_S_NO_ERROR. The final result of the operation MUST be retrieved using the sequences specified in section 3.1.1.1.

The failed method execution MUST set the value referenced by the output parameters to NULL and MUST return an error in the format specified in section 2.2.1.11.

3.1.5.2.19 IWbemServices::ExecQueryAsync (Opnum 21)

The **IWbemServices::ExecQueryAsync** method is the asynchronous version of the **IWbemServices::ExecQuery** method. The **IWbemServices::ExecQueryAsync** method returns an enumerable collection of IWbemClassObject interface objects based on a query.

```
HRESULT ExecQueryAsync(
  [in] const BSTR strQueryLanguage,
  [in] const BSTR strQuery,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [in] IWbemObjectSink* pResponseHandler
);
```

strQueryLanguage: MUST contain one of the query languages supported by WMI. This parameter MUST NOT be NULL.<11>

strQuery: MUST contain the WQL query text as specified in section 2.2.1.1. This parameter MUST NOT be NULL.

lFlags: Flags that affect the behavior of the **IWbemServices::ExecQueryAsync** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_USE_AMENDED_QUALIFIERS 0x00020000	If this bit is not set, the server SHOULD not return CIM Localizable Information. If this bit is set, the server SHOULD return CIM Localizable Information.
WBEM_FLAG_SEND_STATUS 0x00000080	If this bit is not set the server MUST make just one final **IWbemObjectSink::SetStatus** call on the interface pointer provided in pResponseHandler parameter. If this bit is set, the server MAY make intermediate **IWbemObjectSink::SetStatus** calls on the interfaces pointer prior to call completion.
WBEM_FLAG_PROTOTYPE 0x00000002	When this bit is not set, the server SHOULD execute the query. When this bit is set, the server SHOULD NOT execute the query and instead return an object that looks like a typical result object.
WBEM_FLAG_DIRECT_READ 0x00000200	When this bit is not set, the server MUST consider the entire class hierarchy when returning the result. When this bit is set, the server MUST disregard any derived class when searching the result.

The flags MAY be combined in any form, provided that all restrictions provided in the flags description are respected. Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an **IWbemContext** interface, which MUST contain additional information the client wants to pass to the server. If pCtx is NULL, the parameter MUST be ignored.

pResponseHandler: MUST be a pointer to **IWbemObjectSink** implemented by the caller, where enumeration results must be delivered. The parameter MUST NOT be NULL.

In error cases, indicated by the HRESULT return value, the supplied **IWbemObjectSink** interface pointer MUST NOT be used . If WBEM_S_NO_ERROR (as specified in section 2.2.1.11) is returned, the user's **IWbemObjectSink** interface pointer MUST be called to indicate the results of the **ExecQueryAsync** operation, as specified in section 3.1.5.5.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (as specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_ENABLE and WBEM_REMOTE_ENABLE accesses to the namespace.

In response to an **IWbemServices::ExecQueryAsync**, the server MUST evaluate the strQueryLanguage and strQuery parameters (as specified in this section) and return all instances matching the requested query. The method MUST fail if the method parameters or their combinations are not valid as specified earlier in this section, or if the server is unable to execute the method.

The successful asynchronous method execution MUST complete immediately and MUST return WBEM_S_NO_ERROR. The server MUST continue method execution after the WBEM_S_NO_ERROR return and MUST complete the operation using the message sequence specified in section 3.1.1.3.

The failed method execution MUST return an error in the format specified in section 2.2.1.11.

3.1.5.2.20 IWbemServices::ExecNotificationQuery (Opnum 22)

The **IWbemServices::ExecNotificationQuery** method SHOULD be called by the client if the client desires to subscribe for event notifications. In response, the server executes a query to receive events. The call returns immediately, and the user can poll the returned enumerator for events as they arrive. Releasing the returned enumerator cancels the query.

```
HRESULT ExecNotificationQuery(
  [in] const BSTR strQueryLanguage,
  [in] const BSTR strQuery,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [out] IEnumWbemClassObject** ppEnum
);
```

strQueryLanguage: MUST contain one of the query languages supported by WMI. This parameter MUST NOT be NULL.<12>

strQuery: MUST contain the WQL event-related query text as specified in section 2.2.1.1. This parameter MUST NOT be NULL.

lFlags: Flags that affect the behavior of the **IWbemServices::ExecNotificationQuery** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_USE_AMENDED_QUALIFIERS 0x00020000	If this bit is not set, the server SHOULD return no CIM Localizable Information. If this bit is set, the server SHOULD return CIM Localizable Information.
WBEM_FLAG_RETURN_IMMEDIATELY 0x00000010	If this bit is set, the server MUST make the method call semisynchronously. This flag MUST always be set.
WBEM_FLAG_FORWARD_ONLY 0x00000020	When the bit is set, the server MUST return an enumerator without reset capability as specified in section 3.1.5.3. This flag MUST always be set.

The flags MAY be combined in any form, provided that all restrictions provided in the flags description are respected. Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an **IWbemContext** interface, which MUST contain additional information the client wants to pass to the server. If pCtx is NULL, the parameter MUST be ignored.

ppEnum: MUST be an out parameter, which MUST implement the **IEnumWbemClassObject** interface, which MUST receive the pointer to the enumerator that is used to enumerate through the CIM objects returned for the query result set. This parameter MUST NOT be NULL.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (as specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_ENABLE and WBEM_REMOTE_ENABLE accesses to the namespace.

In response to **IWbemServices::ExecNotificationQuery**, the server MUST evaluate the strQuery and strQueryLanguage parameters (as specified in this section) and MUST return all events matching the query. The method MUST fail if the method parameters or their combinations are not valid as specified earlier in this section, or if the server is unable to execute the method. Since the stream of events returned by the server is not finite, the method **IWbemServices::ExecNotificationQuery** MUST NOT be executed synchronously. This request MUST fail as the method parameters are not valid, as specified above.

The successful semisynchronous method execution MUST return immediately, it MUST fill ppEnum parameter with a new **IEnumWbemClassObject** interface pointer and MUST return WBEM_S_NO_ERROR. The final result of the operation MUST be retrieved using the sequences specified in section 3.1.1.1.

The failed method execution MUST set the value referenced by the output parameters to NULL and MUST return an error in the format as specified in section 2.2.1.11.

3.1.5.2.21 IWbemServices::ExecNotificationQueryAsync (Opnum 23)

The **IWbemServices::ExecNotificationQueryAsync** method is the asynchronous version of the **IWbemServices::ExecNotificationQuery** method. The **IWbemServices::ExecNotificationQueryAsync** method SHOULD be called by the client if the client desires to subscribe for asynchronous event notifications. In response, the server performs the same task as the **IWbemServices::ExecNotificationQuery** method, except that events are supplied to the specified response handler (pResponseHandler) until the **IWbemServices::CancelAsyncCall** method SHOULD be called to stop the event notification.

```
HRESULT ExecNotificationQueryAsync(
  [in] const BSTR strQueryLanguage,
  [in] const BSTR strQuery,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [in] IWbemObjectSink* pResponseHandler
);
```

strQueryLanguage: MUST contain one of the query languages supported by WMI. This parameter MUST NOT be NULL. <13>

strQuery: MUST contain the WQL event-related query text as specified in section 2.2.1.1. This parameter MUST NOT be NULL.

lFlags: Flags that affect the behavior of the **IWbemServices::ExecNotificationQueryAsync** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_USE_AMENDED_QUALIFIERS 0x00020000	If this bit is not set, the server SHOULD return no CIM Localizable Information. If this bit is set, the server SHOULD return CIM Localizable Information.
WBEM_FLAG_SEND_STATUS 0x00000080	If this bit is not set the server MUST make just one final **IWbemObjectSink::SetStatus** call on the interface pointer provided in pResponseHandler parameter. If this bit is set, the server MAY make intermediate **IWbemObjectSink::SetStatus** calls on the interfaces pointer prior to call completion.

The flags MAY be combined in any form, provided that all restrictions provided in the flags description are respected. Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an **IWbemContext** interface, which MUST contain additional information the client wants to pass to the server. If pCtx is NULL, the parameter MUST be ignored.

pResponseHandler: MUST be a pointer to **IWbemObjectSink** implemented by the caller, where enumeration results must be delivered. The parameter MUST NOT be NULL. In error cases, indicated by the HRESULT return value, the supplied **IWbemObjectSink** interface pointer MUST NOT be used and the client MAY release it when the call is complete. If WBEM_S_NO_ERROR (as specified in **Protocol Return Codes (section 2.2.1.3)**) is returned,

then the user's **IWbemObjectSink** interface pointer MUST be called to indicate the results of the **ExecNotificationQueryAsync** operation, as specified later in this section.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_ENABLE and WBEM_REMOTE_ENABLE accesses to the namespace.

In response to **IWbemServices::ExecNotificationQueryAsync** the server MUST evaluate the strQueryLanguage and strQuery parameters (as specified earlier in this section) and MUST start providing events matching the requested query. The method MUST fail if the method parameters or their combinations are not valid as specified earlier in this section, or if the server is unable to execute the method.

The successful asynchronous method execution MUST complete immediately and MUST return WBEM_S_NO_ERROR. The server MUST continue method execution after the WBEM_S_NO_ERROR return and MUST complete the operation using the message sequence as specified in section 3.1.1.3.

The failed method execution MUST return an error in the format specified in section 2.2.1.11.

3.1.5.2.22 IWbemServices::ExecMethod (Opnum 24)

The **IWbemServices::ExecMethod** method SHOULD be called to execute a CIM method implemented by a CIM class or a CIM instance retrieved from the **IWbemServices** interface.

```
HRESULT ExecMethod(
  [in] const BSTR strObjectPath,
  [in] const BSTR strMethodName,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [in] IWbemClassObject* pInParams,
  [out, in, unique] IWbemClassObject** ppOutParams,
  [out, in, unique] IWbemCallResult** ppCallResult
);
```

strObjectPath: MUST be the CIM path to the class/instance implementing the method. This parameter MUST NOT be NULL. The CIM path MUST contain the class name and the value of the key properties.

strMethodName: MUST be the name of the method to be executed. This parameter MUST NOT be NULL.

lFlags: Flags that affect the behavior of the **IWbemServices::ExecMethod** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_RETURN_IMMEDIATELY 0x00000010	If this bit is not set, the server MUST make the method call synchronously.

Value	Meaning
	If this bit is set, the server MUST make the method call semisynchronously.

Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an **IWbemContext** interface, which MUST contain additional information the client wants to pass to the server. If pCtx is NULL, the parameter MUST be ignored.

pInParams: MUST be a pointer to an **IWbemClassObject** interface pointer, which MUST contain the values corresponding to all input parameter accepted by the method. This parameter MUST be NULL when the method has no input parameters.

ppOutParams: If ppOutParams is NULL, the parameter MUST be ignored. When the parameter is not NULL, it MUST return an **IWbemClassObject** interface pointer containing output parameters. The output parameter MUST be filled according to the to the state of the **lFlags** parameter (whether WBEM_FLAG_RETURN_IMMEDIATELY is set) as listed in the following table:

Flag state	Success operation	Failure operation
WBEM_FLAG_RETURN_IMMEDIATELY is not set.	contain an IWbemClassObject interface pointer.	MUST be set to NULL if the input parameter is not-NULL.
WBEM_FLAG_RETURN_IMMEDIATELY is set.	MUST return NULL.	MUST be set to NULL if the input parameter is not-NULL.

ppCallResult: This parameter is optional. If the parameter is set to NULL, the server MUST ignore it. However, when the parameter is not NULL, the server MUST return an **IWbemCallResult** interface pointer, which SHOULD be polled to obtain the operation call result. In such a case, the output parameter MUST be filled according to to the state of the **lFlags** parameter (whether WBEM_FLAG_RETURN_IMMEDIATELY is set) as listed in the following table:

Condition	Success operation	Failure operation
WBEM_FLAG_RETURN_IMMEDIATELY is not set.	MUST be set to NULL if the input parameter is not-NULL.	MUST be set to NULL if the input parameter is not-NULL.
WBEM_FLAG_RETURN_IMMEDIATELY is set.	MUST contain the IWbemCallResult interface pointer.	MUST be set to NULL if the input parameter is not-NULL.

Return Values: This method MUST return an HRESULT, which MUST indicate the status of the method call. HRESULT MUST have the type and values as specified in section 2.2.1.11. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_ METHOD EXECUTE and WBEM_REMOTE_ENABLE accesses to the namespace.

In response to **IWbemServices::ExecMethod**, the server MUST evaluate the strObjectPath and strMethodName parameters (as specified in this section) and MUST execute the method identified by strMethodName and implemented by the CIM object referred by strObjectPath. The method MUST use the parameters provided by pInParameters. The method MUST fail if the CIM object referred by strObjectPath does not exist, if the method parameters are not valid as specified earlier in this section, or if the server is unable to execute the method.

The successful synchronous method execution MUST return the output parameters encapsulated in an IWbemClassObject interface pointer in ppObject parameter and MUST return WBEM_S_NO_ERROR.

The successful semisynchronous method execution MUST complete immediately, it MUST fill the ppCallResult with a new **IWbemCallResult** interfaces pointer and MUST return WBEM_S_NO_ERROR. The final result of the operation MUST be retrieved using the sequences specified in section 3.1.5.3.2.

The failed method execution MUST set the output parameters to NULL and MUST return an error in the format as specified in section 2.2.1.11.

3.1.5.2.23 IWbemServices::ExecMethodAsync (Opnum 25)

The **IWbemServices::ExecMethodAsync** method SHOULD be called to asynchronously execute a CIM method implemented by a CIM class or a CIM instance retrieved from the **IWbemServices** interface.

```
HRESULT ExecMethodAsync(
  [in] const BSTR strObjectPath,
  [in] const BSTR strMethodName,
  [in] long lFlags,
  [in] IWbemContext* pCtx,
  [in] IWbemClassObject* pInParams,
  [in] IWbemObjectSink* pResponseHandler
);
```

strObjectPath: MUST be the CIM path to the class/instance implementing the method. This parameter MUST NOT be NULL. The CIM path MUST contain the class name and the value of the key properties.

strMethodName: MUST be the name of the method to be executed. This parameter MUST NOT be NULL.

lFlags: Flags that affect the behavior of the **ExecMethodAsync** method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_SEND_STATUS 0x00000080	If this bit is not set the server MUST make just one final **IWbemObjectSink::SetStatus** call on the interface pointer provided in pResponseHandler parameter. If this bit is set, the server MAY make intermediate **IWbemObjectSink::SetStatus** calls on the interfaces pointer prior to call completion.

Any other DWORD value not matching the above condition MUST be treated as invalid.

pCtx: MUST be a pointer to an **IWbemContext** interface, which MUST contain additional information the client wants to pass to the server. If pCtx is NULL, the parameter MUST be ignored.

pInParams: MUST be a pointer to an **IWbemClassObject** interface pointer, which MUST contain the values of all input parameter accepted by the method. This parameter MUST be NULL when the method has no input parameters.

pResponseHandler: MUST be a pointer to an **IWbemObjectSink** interface object implemented by the client of this method. This parameter MUST NOT be NULL. In error cases, indicated by the HRESULT return value, the supplied **IWbemObjectSink** interface pointer MUST NOT be used. If the return value is equal to WBEM_S_NO_ERROR (as specified in section 2.2.1.11), the user's **IWbemObjectSink** interface pointer MUST be used to indicate the result of the ExecMethodAsync operation, as specified in section 3.1.5.5.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (as specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST have WBEM_ METHOD EXECUTE and WBEM_REMOTE_ENABLE accesses to the namespace.

In response to **IWbemServices::ExecMethodAsync**, the server MUST evaluate the strObjectPath and strMethodName (as specified in this section) and MUST execute the method identified by strMethodName, implemented by strObjectPath CIM object using the provided pInParameters parameters. .The method MUST fail if the method parameters or their combinations are not valid as specified earlier in this section, or if the server is unable to execute the method.

The successful asynchronous method execution MUST complete immediately and MUST return WBEM_S_NO_ERROR. The server MUST continue method execution after the WBEM_S_NO_ERROR return and MUST complete the operation using the message sequence specified in section 3.1.1.3.

The failed method execution MUST return an error in the format specified in section 2.2.1.11.

3.1.5.3 IEnumWbemClassObject Interface

IEnumWbemClassObject interface MUST be used to return results from synchronous and semisynchronous method calls, which can return multiple CIM objects as result. The interface MUST be implemented by the server. The interface MUST be uniquely identified by UUID 027947e1-d731-11ce-a357-000000000001.

Methods in RPC Opnum Order

Method	Description
Reset	Causes the server to reset the enumeration sequence back to the beginning of the collection of CIM objects. Opnum: 3
Next	Causes the server to get one or more CIM objects starting at the current position in an enumeration, and to move the current position by the number of CIM objects in the *uCount*

Method	Description
	parameter. Opnum: 4
NextAsync	Asynchronous version of the **IEnumWbemClassObject::Next** method. Opnum: 5
Clone	Causes the server to make a logical copy of the entire enumerator. Opnum: 6
Skip	Causes the server to move the current position in an enumeration ahead by a specified number of CIM objects. Opnum: 7

An **IEnumWbemClassObject** interface object with reset capability MAY be iterated over multiple times. The iteration MUST restart from the first CIM object after each **IEnumWbemClassObject::Reset** method call. An **IEnumWbemClassObject** interface object MUST be returned by **IWbemServices::CreateClassEnum**, **IWbemServices::CreateInstanceEnum**, **IWbemServices::ExecQuery**, **IWbemServices::ExecNotificationQuery**, as specified in section **IWbemServices**.

The object exporting this interface MUST implement **IWbemFetchSmartEnum** interface. **IRemUnknown** and **IRemUnknown2** interfaces, as specified in [MS-DCOM], MUST be used to manage the interfaces exposed by the object.

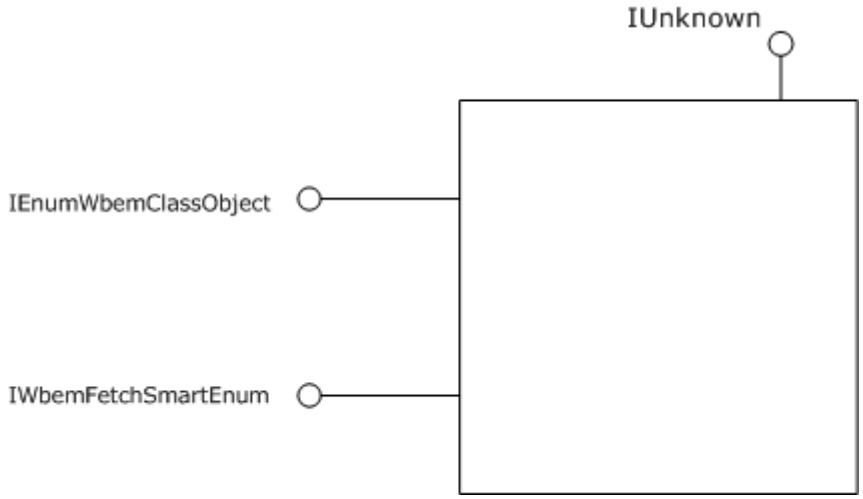


Figure 6: IEnumWbemClassObject interface

3.1.5.3.1 IEnumWbemClassObject::Reset (Opnum 3)

The **IEnumWbemClassObject::Reset** method SHOULD be called by the client. In response, the server MUST reset the enumeration sequence back to the beginning of the collection of CIM objects.

This method has no parameters.

```
HRESULT Reset();
```

This method has no parameters.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method. If the **IEnumWbemClassObject::Reset** method is invoked on an enumerator that does not support the reset capability, the server MUST return WBEM_E_INVALID_OPERATION.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST be the same as the one who obtained the **IEnumWbemClassObject** interface pointer.

In response to the **IEnumWbemClassObject::Reset** method, the server MUST reset the status of the enumeration as specified in this section.

The current index position in the enumeration MUST be reset to 0x0.

The successful method execution MUST return WBEM_S_NO_ERROR.

The failed method execution MUST return an error in the format specified in section 2.2.1.11.

3.1.5.3.2 IEnumWbemClassObject::Next (Opnum 4)

The **IEnumWbemClassObject::Next** method SHOULD be called by the client. In response, the server SHOULD get one or more CIM objects starting at the current position in an enumeration, and MUST move the current position by the number of CIM objects in the *uCount* parameter. When **IEnumWbemClassObject** is created, the current position MUST be set on the first CIM object of the collection. The order of the CIM objects stored in the enumerator is arbitrary.

```
HRESULT Next(
  [in] long lTimeout,
  [in] ULONG uCount,
  [out, size_is(uCount), length_is(*puReturned)]
    IWbemClassObject** apObjects,
  [out] ULONG* puReturned
);
```

lTimeout: MUST be the maximum amount of time, in milliseconds, that the **IEnumWbemClassObject::Next** method call allows to pass before timing out. If the constant WBEM_INFINITE (0xFFFFFFFF) is specified, the call MUST wait until CIM objects are available. If the value 0x0 (WBEM_NO_WAIT) is specified, the call MUST return immediately, regardless of whether any CIM objects are available. This parameter MUST NOT be NULL.

uCount: MUST be the number of requested CIM objects to return. This parameter MUST NOT be NULL.

apObjects: MUST be a pointer to an array of **IWbemClassObject** interface pointer. When sent by the client, this parameter MUST NOT be NULL. Upon return by the server, this parameter can be NULL if there is a failure, or if there are no results.

puReturned: MUST be a pointer to a ULONG type that receives the number of CIM objects returned, which MAY be less than the number requested in uCount. When sent by the client, this parameter MUST NOT be NULL. Upon return by the server, this parameter can be NULL if there is a failure, or if there are no results.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST be the same as the one who obtained **IEnumWbemClassObject** interface pointer.

In response to **IEnumWbemClassObject::Next** method, the server MUST evaluate the uCount and lTimeoutput parameters (as specified in this section) and MUST return the requested number of CIM objects if available. It MUST perform the operation within the lTimeout timeout.

If the server is not able to return all the requested CIM objects in the requested amount of time, it MUST return WBEM_S_TIMEDOUT. The requested number of CIM objects MUST start from the current index position. The current index position in the enumeration MUST be incremented with the number of CIM objects returned.

The successful method execution MUST return WBEM_S_NO_ERROR. If the number of the remaining CIM objects to be retrieved is less than the number of requested CIM objects, the server MUST return WBEM_S_FALSE. In any case, the server MUST fill the output parameters of the method as specified earlier in this section above.

If the original semisynchronous operation fails, the server MUST return the error code that the original method would have returned in its synchronous version.

The failed method execution MUST set the value referenced by the output parameters to NULL and MUST return an error in the format specified in section 2.2.1.11.

3.1.5.3.3 IEnumWbemClassObject::NextAsync (Opnum 5)

The **IEnumWbemClassObject::NextAsync** method is the asynchronous version of the **IEnumWbemClassObject::Next** method. The **IEnumWbemClassObject::NextAsync** method SHOULD be called by the client when controlled asynchronous retrieval of CIM objects to a sink is required. In response, the server asynchronously MUST get one or more CIM objects starting at the current position in an enumeration, and MUST move the current position by the number of CIM objects in the uCount parameter. When **IEnumWbemClassObject** is created, the current position MUST be set on the first CIM object of the collection. The order of the CIM objects stored in the enumerator is arbitrary.

```
HRESULT NextAsync(
  [in] ULONG uCount,
  [in] IWbemObjectSink* pSink
);
```

uCount: MUST be the number of CIM objects being requested. This parameter MUST NOT be NULL.

pSink: MUST be a pointer to the **IWbemObjectSink** interface, that MUST represent the sink to receive the CIM object. As each batch of CIM objects is requested, they MUST be delivered to the Indicate method that pSink points to (as specified in section 3.1.5.5.1), MUST be followed by a final call to the SetStatus method that pSink points to (as specified in section 3.1.5.5.2). This parameter MUST NOT be NULL. In error cases, indicated by the HRESULT return value,

the supplied **IWbemObjectSink** interface pointer MUST NOT be used and the client MAY release it when the call is complete.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST be the same as the one who obtained **IEnumWbemClassObject** interface pointer.

In response to **IEnumWbemClassObject::NextAsync**, the server MUST evaluate the uCount parameter as specified in this section. The lesser of the requested number or the number of CIM objects MUST start from the current index position. If the number of the remaining CIM objects to be retrieved is less than the number of requested CIM objects, the lesser number of CIM objects MUST be returned. The current index position in the enumeration MUST be incremented with the number of CIM objects returned.

The successful method execution MUST return immediately. The server MUST use the provided pSink to report data and status using **IWbemObjectSink::Indicate** specified in **IWbemObjectSink::SetStatus** and sections 3.1.5.5.1 and 3.1.5.5.2.

The failed method execution MUST return an error in the format specified in section 2.2.1.11.

3.1.5.3.4 IEnumWbemClassObject::Clone (Opnum 6)

The **IEnumWbemClassObject::Clone** method SHOULD be called by the client. In response, the server MUST make a logical copy of the entire enumerator. The cloned enumerator MUST have the same current position as the source enumerator.

```
HRESULT Clone(
  [out] IEnumWbemClassObject** ppEnum
);
```

ppEnum: MUST be a pointer to a new **IEnumWbemClassObject** interface CIM object that is a logical copy of the entire enumerator that made the **Clone** method call, retaining the current position in an enumeration. When sent by the client, this parameter MUST NOT be NULL. Upon return by the server, this parameter can be NULL if there is a failure, or if there are no results.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST be the same as the one who obtained **IEnumWbemClassObject** interface pointer.

The successful method execution MUST fill the ppEnum with an **IEnumWbemClassObject** interface pointer, as specified in section 3.1.5.3, which MUST be a copy of the source enumerator retaining the current position in an enumeration. The method MUST return WBEM_S_NO_ERROR.

If the original semisynchronous operation fails, the server MUST return the error code that the original method would have returned in its synchronous version.

The failed method execution MUST return an error in the format specified in section 2.2.1.11.

3.1.5.3.5 IEnumWbemClassObject::Skip (Opnum 7)

The **IEnumWbemClassObject::Skip** method SHOULD be called by the client. In response, the server MUST move the current position in an enumeration ahead by a specified number of CIM objects.

The **IEnumWbemClassObject::Skip** method opnum equals 7.

```
HRESULT Skip(
  [in] long lTimeout,
  [in] ULONG nCount
);
```

lTimeout: MUST be a maximum amount of time in milliseconds that the call to Skip allows to pass before timing out. If the constant WBEM_INFINITE (0xFFFFFFFF) is used, the Skip method call waits until the operation succeeds. This parameter MUST NOT be NULL.

nCount: MUST be the number of CIM objects to skip in the enumeration. If this parameter is greater than the number of CIM objects left to enumerate, the call MUST skip to the end of the enumeration, and WBEM_S_FALSE MUST be returned for the method return value. This parameter MUST NOT be NULL.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The security principal making the call MUST be the same as the one who obtained **IEnumWbemClassObject** interface pointer.

In response to the **IEnumWbemClassObject::Skip** method, the server MUST evaluate the uCount and lTimeoutput parameters as specified in this section. The server MUST skip the requested number of CIM objects from the result set. The server MUST complete the operation within the lTimeout timeout. The requested number of CIM objects MUST start from the current index position. The current index position in the enumeration MUST be incremented with the number of CIM objects skipped.

If the call succeeds, it MUST return WBEM_S_NO_ERROR. If the call gets to the end of the enumeration and the number of the remaining CIM objects is less than the number of requested CIM objects, the server MUST return WBEM_S_FALSE. If the server is not able to skip the requested number of CIM objects in the requested amount of time, it MUST return WBEM_S_TIMEDOUT.

If the original semisynchronous operation fails, the server MUST return the error code that the original method would have returned in its synchronous version.

The failed method execution MUST return an error in the format specified in section 2.2.1.11.

3.1.5.4 IWbemCallResult Interface

The **IWbemCallResult** interface MUST be used to return call results from semisynchronous calls returning a single CIM object. The interface MUST be implemented by the server. The interface MUST be uniquely identified by UUID 44aca675-e8fc-11d0-a07c-00c04fb68820.

Methods in RPC Opnum Order

Method	Description
GetResultObject	Causes the server to attempt to retrieve a CIM object from a previous semisynchronous call to the **IWbemServices::GetObject** or **IWbemServices::ExecMethod** method. Opnum: 3
GetResultString	Causes the server to return the assigned CIM path of a CIM instance that was newly created by the **IWbemServices::PutInstance** method. Opnum: 4
GetResultService	Causes the server to retrieve a pointer to the **IWbemServices** interface that results from a semisynchronous call to the **IWbemServices::OpenNamespace** method. Opnum: 5
GetCallStatus	Causes the server to return the status of the current outstanding semisynchronous call. Opnum: 6

3.1.5.4.1 IWbemCallResult::GetResultObject (Opnum 3)

The **IWbemCallResult::GetResultObject** method SHOULD be called by the client. In response, the server MUST attempt to retrieve a CIM object from a previous semisynchronous call to the **IWbemServices::GetObject** or **IWbemServices::ExecMethod** method.

```
HRESULT GetResultObject(
  [in] long lTimeout,
  [out] IWbemClassObject** ppResultObject
);
```

lTimeout: MUST be a maximum amount of time in milliseconds that the call to **IWbemCallResult::GetResultObject** method allows to pass before timing out. If the constant WBEM_INFINITE (0xFFFFFFFF) is used, the **GetResultObject** method call MUST wait until the operation succeeds. If this parameter is set to 0, the call immediately returns either the object or a status code. If the object is available, then it will return WBEM_S_NO_ERROR, and put a copy of the object in **ppResultObject**. If it is not available at that time, it will return WBEM_S_TIMEDOUT.

ppResultObject: MUST be a copy of the CIM object when the semisynchronous operation is complete. A new CIM object MUST not be returned on error. When sent by the client, this parameter MUST NOT be NULL. Upon return by the server, this parameter can be NULL if there is a failure, or if there are no results.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemCallResult::GetResultObject** method MUST be called on the interface obtained in responses to a previous call to a semisynchronous operation returning an **IWbemCallResult** interface.

In response to the **IWbemCallResult::GetResultObject** method, the server MUST return the CIM object in the *ppResultObject* parameter in the time allowed by *lTimeoutput* parameter. The method MUST fail if the method parameters are not valid as specified earlier in this section, or if the server is unable to execute the method.

The successful method execution MUST fill *ppResultObject* with a new IWbemClassObject interface pointer and MUST return WBEM_S_NO_ERROR.

The failed method execution sets the value referenced by the output parameters to NULL and MUST return an error in the format specified in section 2.2.1.11. In case the operation is not completed after lTimeout milliseconds, the server MUST return WBEM_S_TIMEDOUT and the client MUST retry the operation.

If the original semisynchronous operation fails, the **IWbemCallResult::GetResultObject** method MUST return the error code that the original method would have returned in its synchronous version.

3.1.5.4.2 IWbemCallResult::GetResultString (Opnum 4)

The **IWbemCallResult::GetResultString** method SHOULD be called by the client. In response, the server MUST return the assigned CIM path of a CIM instance that was newly created by the **IWbemServices::PutInstance** method.

```
HRESULT GetResultString(
  [in] long lTimeout,
  [out] BSTR* pstrResultString
);
```

lTimeout: MUST be a maximum amount of time in milliseconds that the call to **GetResultString** allows to pass before timing out. If the constant WBEM_INFINITE (0xFFFFFFFF) is used, the GetResultString method call MUST wait until the operation succeeds. This parameter MUST NOT be NULL.

pstrResultString: MUST be a pointer to a BSTR value, which MUST contain the CIM path of the CIM object instance, which MUST lead to the newly created CIM instance. In case of failure of the semisynchronous operation, the returned string MUST be NULL. When sent by the client, this pointer parameter MUST NOT be NULL. If the original operation does not return a string, the returned string MUST be NULL.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemCallResult::GetResultString** method MUST be called on the interface obtained in responses to a previous call to a semisynchronous operation returning an **IWbemCallResult** interface.

IWbemCallResult::GetResultString MUST be called to obtain the CIM path created after the **IWbemServices::PutInstance** execution. In response to **IWbemCallResult::GetResultString** method, the server MUST return the string result of the operation in pstrResultString parameter. The method MUST fail if the method parameters are not valid as specified earlier in this section, or if the server is unable to execute the method.

The successful method execution MUST fill pstrResultString with a new BSTR and MUST return WBEM_S_NO_ERROR.

The failed method execution sets the value referenced by the output parameters to NULL and MUST return an error in the format specified in section 2.2.1.11. In case the operation is not completed after lTimeout milliseconds, the server MUST return WBEM_S_TIMEDOUT and the client SHOULD retry the operation.

If the original semisynchronous operation fails, the **IWbemCallResult::GetResultString** method MUST return the error code that the original method would have returned in its synchronous version.

3.1.5.4.3 IWbemCallResult::GetResultService (Opnum 5)

The **IWbemCallResult::GetResultServices** method SHOULD be called by the client. In response, the server MUST retrieve a pointer to the **IWbemServices** interface that results from a semisynchronous call to the **IWbemServices::OpenNamespace** method.

GetResultServices method opnum equals 5.

```
HRESULT GetResultService(
  [in] long lTimeout,
  [out] IWbemServices** ppServices
);
```

lTimeout: MUST be a maximum amount of time in milliseconds that the call to **GetResultServices** allows to pass before timing out. If the constant WBEM_INFINITE (0xFFFFFFFF) is used, the Skip method call MUST wait until the operation succeeds. This parameter MUST NOT be NULL.

ppServices: MUST be a pointer to the **IWbemServices** interface requested by the original call to **IWbemServices::OpenNamespace**, when that interface becomes available. In case of failure of the semisynchronous operation, the returned parameter MUST be NULL. When sent by the client, this pointer parameter MUST NOT be NULL. If the original operation does not return an interface pointer, the returned parameter MUST be NULL.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemCallResult::GetResultServices** method MUST be called on the interface obtained in responses to a previous call to a semisynchronous operation returning an **IWbemCallResult** interface.

IWbemCallResult::GetResultServices MUST be called to obtain the **IWbemServices** interface pointer returned by the **IWbemServices::OpenNamespace** execution. In response to the **IWbemCallResult::GetResultServices** method, the server MUST return the IWbemServices interface pointer result of the operation in ppResultServices parameter. The method MUST fail if the method parameters are not valid as specified earlier in this section, or if the server is unable to execute the method.

The successful method execution MUST fill pstrResultString with a new BSTR and MUST return WBEM_S_NO_ERROR.

The failed method execution sets the value referenced by the output parameters to NULL and MUST return an error in the format specified in section 2.2.1.11. In case the operation is not completed after lTimeout milliseconds, the server MUST return WBEM_S_TIMEDOUT and the client SHOULD retry the operation.

If the original semisynchronous operation fails, the **IWbemCallResult::GetResultServices** method MUST return the error code that the original method would have returned in its synchronous version.

3.1.5.4.4 IWbemCallResult::GetCallStatus (Opnum 6)

The **IWbemCallResult::GetCallStatus** method SHOULD be called by the client. In response, the server MUST return the status of the current outstanding semisynchronous call.

```
HRESULT GetCallStatus(
  [in] long lTimeout,
  [out] long* plStatus
);
```

lTimeout: MUST be the maximum amount of time in milliseconds that the call to **GetCallStatus** allows to pass before timing out. If the constant WBEM_INFINITE (0xFFFFFFFF) is used, the Skip method call waits until the operation succeeds. This parameter MUST NOT be NULL.

plStatus: MUST be the status of a call to an **IWbemServices** method if the WBEM_S_NO_ERROR code is returned for this method. When sent by the client, this parameter MUST NOT be NULL. Upon return by the server, this parameter can be NULL if there is a failure, or if there are no results.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemCallResult::GetCallStatus** method MUST be called on the interface obtained in responses to a previous call to a semisynchronous operation returning an **IWbemCallResult** interface.

In response to an **IWbemCallResult::GetCallStatus** method, the server MUST return operation status in plStatus parameter in the time allowed by lTimeoutput parameter. The method MUST fail if

the method parameters are not valid as specified earlier in this section, or if the server is unable to execute the method.

The successful method execution MUST fill plStatus with a new BSTR and MUST return WBEM_S_NO_ERROR. In case the operation is not completed after lTimeout milliseconds, the server MUST return WBEM_S_TIMEDOUT and the client SHOULD retry the operation.

The failed method execution sets the value referenced by the output parameters to NULL and MUST return an error in the format specified in section 2.2.1.11.

3.1.5.5 IWbemObjectSink Interface

The **IWbemObjectSink** interface MUST be implemented by the client if the client uses asynchronous method calls as specified in section 3.1.5.2. The interface MUST be invoked by the server when the client requests asynchronous method call results. The server MAY call the **IWbemObjectSink::Indicate** method to provide CIM objects to the client, after which the server MUST call the **IWbemObjectSink::SetStatus** method to indicate the end of the notification sequence. The **IWbemObjectSink** interface is a Distributed Component Object Model (DCOM) Remote Protocol (as specified in [MS-DCOM]) interface. The interface MUST be uniquely identified by UUID 7c857801-7381-11cf-884d-00aa004b2e24.

Methods in RPC Opnum Order

Method	Description
Indicate	Called by the server to return additional results. Opnum: 3
SetStatus	Called by the server either to indicate the end of an operation or to send status information to the client. Opnum: 4

3.1.5.5.1 IWbemObjectSink::Indicate (Opnum 3)

The **IWbemObjectSink::Indicate** method MUST be called by the server to return additional results. The **IWbemObjectSink::Indicate** method has the following syntax, expressed in **MIDL** language:

```
HRESULT Indicate(
  [in] long lObjectCount,
  [in, size_is(lObjectCount)] IWbemClassObject** apObjArray
);
```

lObjectCount: MUST be the number of CIM objects in the array of pointers in the ppObjArray parameter. This parameter MUST NOT be NULL.

apObjArray: MUST be an array of **IWbemClassObject** <14> interface pointers.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The client MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The server MUST call **IWbemObjectSink::Indicate** method on the interface provided as response handler on all IWbemServices asynchronous methods.

The server calls **IWbemObjectSink::Indicate** multiple times until the entire result set is delivered to the client. If no results are reported, the server MUST NOT call this method.

If the **IWbemObjectSink::Indicate** method fails, the server MUST cancel the current operation.

The successful method execution MUST return WBEM_S_NO_ERROR.

Optimization

A client able to handle an ObjectArray structure <15> MUST notify the server by returning 0x400FF (WBEM_S_NEW_STYLE) in response to the first **IWbemObjectSink::Indicate** call using the DCOM Remote Protocol marshaling.

Clients supporting ObjectArray structure MUST validate if the result set is sent using the DDCOM Remote Protocol marshaling or using the ObjectArray structure. These clients MUST accept both formats.

Servers supporting ObjectArray structure MUST detect the WBEM_S_NEW_STYLE and MUST send the rest of the results using the ObjectArray structure as specified in section 3.1.1.3.

3.1.5.5.2 IWbemObjectSink::SetStatus (Opnum 4)

The **IWbemObjectSink::SetStatus** method MUST be called by the server either to indicate the end of an operation or to send status information to the client.

```
HRESULT SetStatus(
  [in] long lFlags,
  [in] HRESULT hResult,
  [in] BSTR strParam,
  [in] IWbemClassObject* pObjParam
);
```

lFlags: Flags that give information about the operation status. The flags MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_STATUS_COMPLETE 0x00000000	Indicates the end of the asynchronous operation.
WBEM_STATUS_PROGRESS 0x00000002	Indicates the progress state of the asynchronous operation.

Any other DWORD value not matching the above condition MUST be treated as invalid.

hResult: The hResult value of the asynchronous operation or notification. This hResult MUST be the same HRESULT that the client gets from the matching synchronous operation.

strParam: MUST be a BSTR value that MUST represent the operational result of the asynchronous operation, when the result of the operation is a string and MUST be NULL otherwise.

pObjParam: MUST be an **IWbemClassObject** interface pointer, which MUST contain additional error information when the asynchronous operation fails.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The client MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The server MUST call the **IWbemObjectSink::SetStatus** method on the interface provided as response handler on all **IWbemServices** asynchronous methods.

The server MUST call **IWbemObjectSink::SetStatus** at the completion of the asynchronous operation passing WBEM_STATUS_COMPLETE as lFlags parameter and the operation return code as HRESULT parameter.

The server MAY call **IWbemObjectSink::SetStatus** multiple times during the operation execution to report the operation progress<16>, as explicitly requested by the client using WBEM_SEND_STATUS flag. In this case, HRESULT parameter MUST contain the progress information.

If the **IWbemObjectSink::SetStatus** method fails, the server MUST cancel the current operation. The successful method execution MUST return WBEM_S_NO_ERROR.

3.1.5.6 IWbemFetchSmartEnum Interface

The **IWbemFetchSmartEnum** interface (an [MS-DCOM] interface) is a helper interface used to retrieve a network-optimized enumerator interface. The server MUST fail the IRemUnknown::QueryInterface operation if the interface is not implemented by the server.

The client SHOULD <17> use this interface when available.

The **IWbemFetchSmartEnum** is a DCOM Remote Protocol interface. The interface MUST be uniquely identified by UUID 1C1C45EE-4395-11d2-B60B-00104B703EFD.

Methods in RPC Opnum Order

Method	Description
GetSmartEnum	Retrieves an **IWbemWCOSmartEnum** interface, which is a network-optimized enumerator interface. Opnum: 3

3.1.5.6.1 IWbemFetchSmartEnum::GetSmartEnum (Opnum 3)

The **IWbemFetchSmartEnum::GetSmartEnum** method retrieves an **IWbemWCOSmartEnum** interface, which is a network-optimized enumerator interface.

```
HRESULT GetSmartEnum(
  [out] IWbemWCOSmartEnum** ppSmartEnum
);
```

ppSmartEnum: MUST be a pointer to a pointer to a network-optimized enumerator interface. When sent by the client, this parameter MUST NOT be NULL. Upon return by the server, this parameter can be NULL if there is a failure, or if there are no results.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemFetchSmartEnum::GetSmartEnum** MUST<18> be called on an interface previously obtained using **IRemUnknown** and **IRemUnknown2** interfaces, as specified in [MS-DCOM], on an **IEnumWbemClassObject** interface previously obtained from calls to the following:

- **IWbemServices::ExecQuery**
- **IWbemServices::CreateInstanceEnum**
- **IWbemServices::CreateClassEnum**
- **IWbemServices::ExecNotificationQuery**
- **IEnumWbemClassObject::Clone**

The security principal making the call MUST be the same as the one who obtained IWbemClassObject interface pointer.

In response to **IWbemFetchSmartEnum::GetSmartEnum** method, the server MUST return an **IWbemWCOSmartEnum** (section 3.1.5.7) interface in the ppSmartEnum output parameter.

A successful execution MUST return the **IWbemWCOSmartEnum** interface in the output parameter and MUST return WBEM_S_NO_ERROR.

The failed method execution MUST set the output parameters to NULL and MUST return an error in the format specified in section 2.2.1.11.

3.1.5.7 IWbemWCOSmartEnum Interface

The server MUST implement the **IWbemWCOSmartEnum** interface if it implements **IWbemFetchSmartEnum::GetSmartEnum**. The **IWbemWCOSmartEnum** interface is intended to provide an alternate synchronous enumeration of CIM objects for **IEnumWbemClassObject**. The client MUST use the **IWbemWCOSmartEnum** interface when available.

The interface MUST be uniquely identified by UUID 423EC01E-2E35-11d2-B604-00104B703EFD.

Methods in RPC Opnum Order

Method	Description
Next	Returns an array of IWbemClassObject interface pointers encoded using ObjectArray structure for optimization purposes. Opnum: 3

3.1.5.7.1 IWbemWCOSmartEnum::Next (Opnum 3)

The **IWbemWCOSmartEnum::Next** method MUST return an array of IWbemClassObject interface pointers encoded using ObjectArray structure for optimization purposes. The array of objects returned in the ObjectArray structure MUST be identical to the array of CIM objects returned by **IWbemWCOSmartEnum::Next**.

```
HRESULT Next(
  [in] REFGUID proxyGUID,
  [in] long lTimeout,
  [in] ULONG uCount,
  [out] ULONG* puReturned,
  [out] ULONG* pdwBuffSize,
  [out, size_is(,*pdwBuffSize)] byte** pBuffer
);
```

proxyGUID: MUST be a client generated GUID which MUST identify the client. This parameter MUST NOT be NULL.

lTimeout: MUST be the maximum amount of time, in milliseconds, that the Next method call allows to pass before timing out. If the constant WBEM_INFINITE (0xFFFFFFFF) is used, the Skip method call waits until the operation succeeds. This parameter MUST NOT be NULL.

uCount: MUST be the number of requested CIM objects. This parameter MUST NOT be NULL.

puReturned: MUST be a pointer to a ULONG value, which MUST contain the number of CIM objects returned by the **Next** method. This parameter MUST NOT be NULL.

pdwBuffSize: MUST be a pointer to an ULONG value that MUST contain the size of the buffer, in bytes. This parameter MUST NOT be NULL.

pBuffer: MUST be a pointer to the byte array which MUST represent the packet. This parameter MUST NOT be NULL. The byte array represents an array of CIM objects encoded using ObjectArray format as specified in section 2.2.1.15.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemWCOSmartEnum::Next** method MUST be called on an **IWbemWCOSmartEnum** interface returned by a previous call to **IWbemFetchSmartEnum::GetSmartEnum**.

The security principal making the call MUST be the same as the one who obtained **IEnumWbemClassObject** interface pointer.

In response to **IWbemWCOSmartEnum::Next**, the server MUST evaluate the lTimeoutput parameter (parameters specified in this section) and MUST evaluate the GUID in order to identify the client. The server MUST return the maximum number of CIM objects requested by uCount.

If the server is not able to return all the requested CIM objects in the requested amount of time, it MUST return WBEM_S_TIMEDOUT. The requested number of CIM objects MUST start from the current index position. The current index position in the enumeration MUST be incremented with the number of CIM objects returned.

Upon successful execution the server MUST return data in the pBuffer using an ObjectArray structure as specified in section 2.2.1.15.

The successful method execution MUST return WBEM_S_NO_ERROR. If the number of the remaining CIM objects to be retrieved is less than the number of requested CIM objects, the server MUST return WBEM_S_FALSE. In any case, the server MUST fill the output parameters of the method as specified in section 2.2.1.15.

3.1.5.8 IWbemLoginClientID Interface

The **IWbemLoginClientID** interface (an [MS-DCOM] interface) MUST <19> be used by the client to send client information to the server, which is optional. The server MUST fail the IRemUnknown::QueryInterface operation if the interface is not implemented by the server and the client SHOULD ignore the error. This interface is not required for the protocol to work.

The interface MUST be uniquely identified by UUID d4781cd6-e5d3-44df-ad94-930efe48a887.

Methods in RPC Opnum Order

Method	Description
SetClientInfo	Passes the client NETBIOS name and a unique client generated number to the server. Opnum: 3

3.1.5.8.1 IWbemLoginClientID::SetClientInfo (Opnum 3)

The **IWbemLoginClientID::SetClientInfo** method MUST pass the client NETBIOS name and a unique client generated number to the server.

```
HRESULT SetClientInfo(
  [in, unique, string] LPWSTR wszClientMachine,
  [in] long lClientProcId,
  [in] long lReserved
);
```

wszClientMachine: MUST specify the client NETBIOS name. This parameter MUST NOT be NULL.

lClientProcId: MUST specify a unique client generated number.

lReserved: This parameter is not used. This parameter MUST be NULL.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemLoginClientID::SetClientInfo** method MAY <20>be called on an interface previously obtained using **IRemUnknown** and **IRemUnknown2** interfaces, as specified in [MS-DCOM], on an IWbemLevel1Login interface previously obtained from DCOM Remote Protocol activation.

The successful method execution MUST return WBEM_S_NO_ERROR.

3.1.5.9 IWbemLoginHelper Interface

The **IWbemLoginHelper** interface (an [MS-DCOM] interface) MAY <21> be used by the client to detect if the server runs on the same system as the client. The server MUST fail the IRemUnknown::QueryInterface operation if the interface is not implemented by the server and the client SHOULD ignore the error. This interface is not required for the protocol to work.

The interface MUST be uniquely identified by UUID 541679AB-2E5F-11d3-B34E-00104BCC4B4A.

Methods in RPC Opnum Order

Method	Description
SetEvent	Signals an event on the server with name that MUST be specified as a parameter of the method. Opnum: 3

3.1.5.9.1 IWbemLoginHelper::SetEvent (Opnum 3)

The **IWbemLoginHelper::SetEvent** signals an event on the server with name that MUST be specified as a parameter of the method. The **SetEvent** method definition is as follows:

The **SetEvent** method opnum equals 3.

```
HRESULT SetEvent(
  [in] const char* sEventToSet
);
```

sEventToSet: MUST be a pointer to a string, which MUST contain the name of the event to be signaled. This parameter MUST NOT be NULL.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemLoginHelper::SetEvent** method MAY <22> be called on an interface previously obtained using **IRemUnknown** and **IRemUnknown2** interfaces, as specified in [MS-DCOM], on an **IWbemLevel1Login** interface previously obtained from DCOM Remote Protocol activation.

The successful method execution MUST return WBEM_S_NO_ERROR.

3.1.5.10 IWbemBackupRestore Interface

The **IWbemBackupRestore** interface exposes methods that back up and restore the contents of the CIM database. The interface MUST be implemented by the server to support backup/restore scenarios. The interface MUST be uniquely identified by UUID C49E32C7-BC8B-11d2-85D4-00105A1F8304.

Methods in RPC Opnum Order

Method	Description
Backup	Causes the server to back up the contents of the CIM database. Opnum: 3
Restore	Causes the server to restore the contents of the CIM database. Opnum: 4

The object exporting this interface MUST also implements the **IWbemBackupRestoreEx** interface. **IRemUnknown** and **IRemUnknown2** interfaces, as specified in [MS-DCOM], MUST be used to manage the interfaces exposed by the object. The object MUST be uniquely identified with CLSID {C49E32C6-BC8B-11D2-85D4-00105A1F8304}.

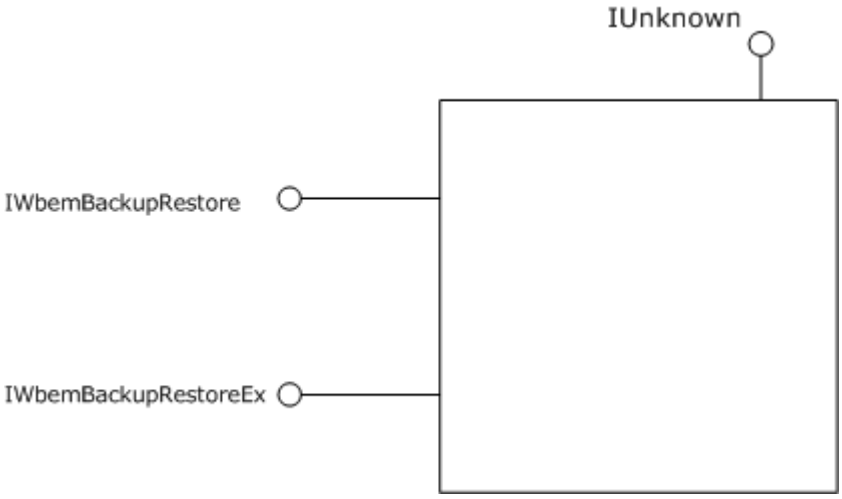


Figure 7: The IWbemBackupRestore interface

3.1.5.10.1 IWbemBackupRestore::Backup (Opnum 3)

On the **IWbemBackupRestore::Backup** method invocation, the server MUST backup the contents of the CIM database.

```
HRESULT Backup(
  [in, string] LPCWSTR strBackupToFile,
  [in] long lFlags
);
```

strBackupToFile: MUST be a UTF-16 string, which MUST contain the name of the file to back up the CIM database to. This parameter MUST NOT<23> be NULL.

lFlags: This parameter is not used, and it MUST be 0x0.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemBackupRestore::Backup** method MUST be called on the interface obtained from DCOM Remote Protocol activation of CLSID_WbemBackupRestore interface as specified in this section.

In response to the **IWbemBackupRestore::Backup** method, the server MUST backup the CIM database in a file specified in strBackupToFile parameter. The server requires the client of the **IWbemBackupRestore::Backup** method to have backup privilege.

The successful method execution MUST return WBEM_S_NO_ERROR.

3.1.5.10.2 IWbemBackupRestore::Restore (Opnum 4)

On the **IWbemBackupRestore::Restore** method invocation, the server MUST restore the contents of the CIM database.

```
HRESULT Restore(
  [in, string] LPCWSTR strRestoreFromFile,
  [in] long lFlags
);
```

strRestoreFromFile: : It MUST be a UTF-16 string, which MUST contain the name of the file to restore the CIM database from. This parameter MUST NOT<24> be NULL.

lFlags: Flags that affect the behavior of the Restore method. The flags behavior MUST be interpreted as specified in the following table.

Value	Meaning
WBEM_FLAG_BACKUP_RESTORE_FORCE_SHUTDOWN 0x00000001	If the bit is not set, the server MUST NOT perform the restore if there are any active clients. If the bit is set, the server MUST shut down any active clients before performing the restore operation.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemBackupRestore::Restore** method MUST be called on the interface obtained from DCOM Remote Protocol activation of CLSID_WbemBackupRestore interface as specified in this section.

In response to **IWbemBackupRestore::Restore** method, the server MUST restore the CIM database from the file specified in strRestoreFrom File parameter. The server requires that the client of the **IWbemBackupRestore::Restore** method MUST have restore privilege.

The successful method execution MUST return WBEM_S_NO_ERROR.

3.1.5.11 IWbemBackupRestoreEx Interface

The **IWbemBackupRestoreEx** interface extends **IWbemBackupRestoreEx** interface and <25> exposes methods that pause and resume the activity in the Windows Management Instrumentation Remote Protocol. These methods are used to provide an alternate solution for backing up the contents of the CIM database. The interface MUST be implemented in order to support backup/restore scenarios without stopping the server.

The **IWbemBackupRestoreEx** interface is a DCOM Remote Protocol interface (as specified in [MS-DCOM]). The interface MUST be uniquely identified by UUID A359DEC5-E813-4834-8A2A-BA7F1D777D76.

Methods in RPC Opnum Order

Method	Description
Pause	Causes the server to lock the CIM database in a consistent state while it is copied. Opnum: 5
Resume	Causes the server to unlock the CIM database and resume operations. Opnum: 6

3.1.5.11.1 IWbemBackupRestoreEx::Pause (Opnum 5)

On the **IWbemBackupRestoreEx::Pause** method invocation, the server MUST lock the CIM database in a consistent state while it is copied.

```
HRESULT Pause();
```

This method has no parameters.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemBackupRestoreEx::Pause** method MUST be called on the interface obtained from the DCOM Remote Protocol activation of the **CLSID_WbemBackupRestore** interface, as specified in this section.

In response to the **IWbemBackupRestoreEx::Pause** method, the server MUST lock the CIM database until the **IWbemBackupRestoreEx::Resume** MUST be called or the Backup timer expires.

The successful method execution MUST return WBEM_S_NO_ERROR.

3.1.5.11.2 IWbemBackupRestoreEx::Resume (Opnum 6)

On the **IWbemBackupRestoreEx::Resume** method invocation, the server MUST unlock the CIM database and resume operations.

```
HRESULT Resume();
```

This method has no parameters.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemBackupRestoreEx::Resume** method MUST be called on the interface obtained from the DCOM Remote Protocol activation of CLSID_WbemBackupRestore interface as specified in this section.

In response to **IWbemBackupRestoreEx::Pause** method, the server MUST unlock the CIM database.

The successful method execution MUST return WBEM_S_NO_ERROR.

3.1.5.12 IWbemRefreshingServices Interface

The **IWbemRefreshingServices** interface (an [MS-DCOM] interface) <26> provides methods that allow clients to get updates of unrelated objects in a single DCOM Remote Protocol method invocation.

The **IWbemRefreshingServices** interface MUST be implemented by the server and MUST be used to configure high speed CIM instance refreshing service.

The interface MUST be uniquely identified by UUID 2C9273E0-1DC3-11d3-B364-00105A1F8177.

Methods in RPC Opnum Order

Method	Description
AddObjectToRefresher	Adds a CIM instance to the list of CIMs that MAY be refreshed. Opnum: 3
AddObjectToRefresherByTemplate	Adds a CIM instance identified by its CIM object instance to the list of CIMs that MAY be refreshed. Opnum: 4
AddEnumToRefresher	Add all CIM instances of the CIM class name to the list of CIMs that MAY be refreshed. Opnum: 5
RemoveObjectFromRefresher	Remove a CIM instance from the list of CIM instances that MAY be refreshed. Opnum: 6
GetRemoteRefresher	Retrieves an **IWbemRemoteRefresher** interface pointer that the client MAY use to refresh objects and enumerations. Opnum: 7
ReconnectRemoteRefresher	Restore a set of CIM instances and enumerations to a server

Method	Description
	refresher. Opnum: 8

3.1.5.12.1 IWbemRefreshingServices::AddObjectToRefresher (Opnum 3)

The **IWbemRefreshingServices::AddObjectToRefresher** method MUST add a CIM instance identified by its CIM path, to the list of CIM instances that can be refreshed.

```
HRESULT AddObjectToRefresher(
  [in] _WBEM_REFRESHER_ID* pRefresherId,
  [in, string] LPCWSTR wszPath,
  [in] long lFlags,
  [in] IWbemContext* pContext,
  [in] DWORD dwClientRefrVersion,
  [out] _WBEM_REFRESH_INFO* pInfo,
  [out] DWORD* pdwSvrRefrVersion
);
```

pRefresherId: MUST be a pointer to the _WBEM_REFRESHER_ID structure, as specified in section 2.2.1.21,which MUST identify the client that is requesting refreshing services. This parameter MUST NOT be NULL.

wszPath: MUST be a string, which MUST contain the CIM path of the CIM instance. This parameter MUST NOT be NULL.

lFlags: This parameter is not used, and it MUST be 0x0.

pContext: MUST be a pointer to an **IWbemContext** interface object, which MUST contain additional information for the server refresher. If pContext is NULL, the parameter MUST be ignored.

dwClientRefrVersion: MUST be the version of the client refresher. It MUST be 0x2.

pInfo: MUST be an output parameter which MUST return a _WBEM_REFRESH_INFO structure specified in section 2.2.1.20,which MUST contain refresher information about CIM instance in wszPath. It MUST NOT be NULL.

pdwSvrRefrVersion: MUST be an output parameter which MUST be the version of the server refresher. It MUST be 0x1.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemRefreshingServices::AddObjectToRefresher** method MUST be called on an interface previously obtained using **IRemUnknown** and **IRemUnknown2** interfaces, as specified in [MS-DCOM], on the **IWbemServices** interface previously obtained in responses to a previous call to **IWbemLevel1Login::NTLMLogin** or **IWbemServices::OpenNamespace**.

In response to **IWbemRefreshingServices::AddObjectToRefresher**, the server MUST evaluate the CIM path to the CIM instance and it MUST return information to the client to handle the given CIM instance as specified in this section.

A successful call into **IWbemRefreshingServices::AddObjectToRefresher** MUST return WBEM_S_NO_ERROR and MUST fill the output **_WBEM_REFRESH_INFO** structure as specified in section 2.2.1.20.

If the wszPath parameter points to an instance of a class derived from CIM_StatisticalInformation, as specified in [DMTF-DSP004], the server MUST return a _WBEM_REFRESH_INFO structure having m_lType member equal to WBEM_REFRESH_TYPE_REMOTE.

In such a case, the _WBEM_REFRESH_INFO_REMOTE structure returned as part of the _WBEM_REFRESH_INFO structure contains an **IWbemRemoteRefresher** interface pointer which MUST be used to invoke **IWbemRemoteRefresher::RemoteRefresh** method.

However, if the wszPath parameter points to an instance of a class not derived from CIM_StatisticalInformation, the server MUST return a _WBEM_REFRESH_INFO structure having m_lType member equal to WBEM_REFRESH_TYPE_NON_HIPERF.

In such a case, the client MUST retrieve the CIM instance using the **IWbemServices::GetObject** or **IWbemServices::GetObjectAsync** on the initial **IWbemServices** interface.

In case of failure, the server MUST fill in _WBEM_REFRESH_INFO parameter with 0x0, set its m_lType member to WBEM_REFRESH_TYPE_INVALID and return an HRESULT error in the format specified in section 2.2.1.11.

3.1.5.12.2 IWbemRefreshingServices::AddObjectToRefresherByTemplate (Opnum 4)

The **IWbemRefreshingServices::AddObjectToRefresherByTemplate** method MUST add a CIM instance identified by its CIM object instance, to the list of CIM instances that MAY be refreshed.

The **AddObjectToRefresherByTemplate** method opnum equals 4.

```
HRESULT AddObjectToRefresherByTemplate(
  [in] _WBEM_REFRESHER_ID* pRefresherId,
  [in] IWbemClassObject* pTemplate,
  [in] long lFlags,
  [in] IWbemContext* pContext,
  [in] DWORD dwClientRefrVersion,
  [out] _WBEM_REFRESH_INFO* pInfo,
  [out] DWORD* pdwSvrRefrVersion
);
```

pRefresherId: MUST be a pointer to the _WBEM_REFRESHER_ID structure, as specified in section 2.2.1.21, identifying the client that is requesting refreshing services. This parameter MUST NOT be NULL.

pTemplate: MUST be a pointer to an **IWbemClassObject** interface CIM instance which MUST be a template for the CIM instances that MAY be refreshed by the refresher. This parameter MUST NOT be NULL.

lFlags: This parameter is not used, and it MUST be 0x0.

pContext: MUST be a pointer to an IWbemContext interface object, which MUST contain additional information for the server refresher. If pContext is NULL, the parameter MUST be ignored.

dwClientRefrVersion: MUST be the version of the client refresher. It MUST be 0x2.

pInfo: MUST be an output parameter returning a _WBEM_REFRESH_INFO structure specified in section 2.2.1.20, which MUST contain refresher information about CIM instance in wszPath. It MUST NOT be NULL.

pdwSvrRefrVersion: MUST an output parameter, which MUST be the version of the server refresher. It MUST be 0x1.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemRefreshingServices::AddObjectToRefresherByTemplate** method MUST be called on the interface obtained using **IRemUnknown** and **IRemUnknown2** interfaces, as specified in [MS-DCOM], on the **IWbemServices** interface previously obtained in responses to a previous call to **IWbemLevel1Login::NTLMLogin** or **IWbemServices::OpenNamespace**.

In response to **IWbemRefreshingServices::AddObjectToRefresherByTemplate**, the server MUST evaluate the pTemplate parameter defining the CIM instance, and it MUST return information to the client to handle the given CIM instance as specified in this section.

A successful call into **IWbemRefreshingServices::AddObjectToRefresherByTemplate** MUST return WBEM_S_NO_ERROR and MUST fill the output _WBEM_REFRESH_INFO structure, as specified in this section.

If the pTemplate parameter is an instance of a class derived from CIM_StatisticalInformation, as specified in [DMTF-DSP004], the server MUST return a _WBEM_REFRESH_INFO structure having m_lType member equal to WBEM_REFRESH_TYPE_REMOTE.

In such a case, the _WBEM_REFRESH_INFO_REMOTE structure returned as part of the _WBEM_REFRESH_INFO structure contains an **IWbemRemoteRefresher** interface pointer which MUST be used to invoke **IWbemRemoteRefresher::RemoteRefresh** method.

However, if the pTemplate parameter is an instance of a class not derived from CIM_StatisticalInformation, the server MUST return a _WBEM_REFRESH_INFO structure having m_lType member equal to WBEM_REFRESH_TYPE_NON_HIPERF.

In such a case, the client MUST retrieve the CIM instance using the **IWbemServices::GetObject** or **IWbemServices::GetObjectAsync** on the initial **IWbemServices** interface.

In case of failure, the server MUST fill in _WBEM_REFRESH_INFO parameter with 0x0, set its m_lType to WBEM_REFRESH_TYPE_INVALID and MUST return an error in the format specified in section 2.2.1.11.

3.1.5.12.3 IWbemRefreshingServices::AddEnumToRefresher (Opnum 5)

The **IWbemRefreshingServices::AddEnumToRefresher** method MUST add all CIM instances identified by the CIM class name to the list of CIM instances that MAY be refreshed.

```
HRESULT AddEnumToRefresher(
  [in] _WBEM_REFRESHER_ID* pRefresherId,
  [in, string] LPCWSTR wszClass,
  [in] long lFlags,
  [in] IWbemContext* pContext,
  [in] DWORD dwClientRefrVersion,
  [out] _WBEM_REFRESH_INFO* pInfo,
  [out] DWORD* pdwSvrRefrVersion
);
```

pRefresherId: MUST be a pointer to the _WBEM_REFRESHER_ID structure, as specified in section 2.2.1.21, identifying the client that is requesting refreshing services. This parameter MUST NOT be NULL.

wszClass: MUST a string which MUST contain the enumeration CIM class name. This parameter MUST NOT be NULL.

lFlags: This parameter is not used, and it MUST be 0x0.

pContext: MUST a pointer to an **IWbemContext** interface object, which MUST contain additional information for the server refresher. If pContext is NULL, the parameter is ignored.

dwClientRefrVersion: MUST be the version of the client refresher. It MUST be 0x2.

pInfo: MUST be an output parameter returning a _WBEM_REFRESH_INFO structure, as specified in section 2.2.1.20, which MUST contain refresher information about the CIM instance in wszPath. It MUST NOT be NULL.

pdwSvrRefrVersion: MUST be an output parameter, which MUST be the version of the server refresher. It MUST be 0x1.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemRefreshingServices::AddEnumToRefresher** method MUST be called on the interface obtained using the **IRemUnknown** and **IRemUnknown2** interfaces, as specified in [MS-DCOM], on the **IWbemServices** interface that MUST have been previously obtained in responses to a previous call to **IWbemLevel1Login::NTLMLogin** or **IWbemServices::OpenNamespace** method.

In response to **IWbemRefreshingServices::AddEnumToRefresher** the server MUST evaluate the wszClass parameter and it MUST return information to the client so the server knows how to handle the given class as specified in this section.

This method MUST add all instances of a class as opposed to one single instance in case of **IWbemRefreshingServices::AddObjectToRefresher** and **IWbemRefreshingServices::AddObjectToRefresherByTemplate**.

A successful call into **IWbemRefreshingServices::AddEnumToRefresher** MUST return WBEM_S_NO_ERROR and MUST fill the output _WBEM_REFRESH_INFO structure as specified in section 2.2.1.20.

If the wszClass parameter is a class derived from CIM_StatisticalInformation, as specified in [DMTF-DSP004], the server MUST return a _WBEM_REFRESH_INFO structure having m_lType member equal to WBEM_REFRESH_TYPE_REMOTE.

In such a case, the _WBEM_REFRESH_INFO_REMOTE structure returned as part of the _WBEM_REFRESH_INFO structure contains an **IWbemRemoteRefresher** interface pointer which MUST be used to invoke **IWbemRemoteRefresher::RemoteRefresh** method.

However, if the wszClass parameter is a class not derived from CIM_StatisticalInformation, the server MUST return a _WBEM_REFRESH_INFO structure having m_lType member equal to WBEM_REFRESH_TYPE_NON_HIPERF.

In such a case, the client MUST retrieve the CIM instance using the **IWbemServices::GetObject** or **IWbemServices::GetObjectAsync** on the initial **IWbemServices** interface.

In case of failure the server MUST fill in _WBEM_REFRESH_INFO parameter with 0x0, set its m_lType to WBEM_REFRESH_TYPE_INVALID and MUST return an error in the format specified in section 2.2.1.11.

3.1.5.12.4 IWbemRefreshingServices::RemoveObjectFromRefresher (Opnum 6)

The **IWbemRefreshingServices::RemoveObjectFromRefresher** method MUST remove a CIM instance identified by its CIM path, from the list of CIM instances that can be refreshed.

```
HRESULT RemoveObjectFromRefresher(
  [in] _WBEM_REFRESHER_ID* pRefresherId,
  [in] long lId,
  [in] long lFlags,
  [in] DWORD dwClientRefrVersion,
  [out] DWORD* pdwSvrRefrVersion
);
```

pRefresherId: MUST be a pointer to the _WBEM_REFRESHER_ID structure as specified in section 2.2.1.21, that identifies the client that is requesting refreshing services. This parameter MUST NOT be NULL.

lId: This parameter MUST be an identifier to the object that is being removed. This parameter MUST NOT be NULL.

lFlags: This parameter is not used, and it MUST be 0x0.

dwClientRefrVersion: MUST be the version of the client refresher. It MUST be 0x2.

pdwSvrRefrVersion: MUST be an output parameter, which MUST be the version of the server refresher. It MUST be 0x1.

Return Values: This method MUST return an **HRESULT** value that MUST indicate the status of the method call. If there are no failures, the server MUST always return WBEM_E_NOT_AVAILABLE <27>.

WBEM_E_NOT_AVAILABLE (0x80041009)

The **IWbemRefreshingServices::RemoveObjectFromRefresher** method MUST be called on the interface obtained using the **IRemUnknown** and **IRemUnknown2** interfaces, as specified in [MS-DCOM], on the **IWbemServices** interface that MUST have been previously obtained in responses to

a previous call to **IWbemLevel1Login::NTLMLogin** or **IWbemServices::OpenNamespace** method.

In response to **IWbemRefreshingServices::RemoveObjectFromRefresher** the server MUST set **pdwSvrRefrVersion** to 0x1 and return WBEM_E_NOT_AVAILABLE

In case of failure the server DOES NOT modify **pdwSvrRefrVersion** and MUST return an error in the format specified in section 2.2.1.11.

3.1.5.12.5 IWbemRefreshingServices::GetRemoteRefresher (Opnum 7)

The **IWbemRefreshingServices::GetRemoteRefresher** method MUST return an **IWbemRemoteRefresher** interface pointer that the client MAY use to refresh objects and enumerations.

```
HRESULT GetRemoteRefresher(
  [in] _WBEM_REFRESHER_ID* pRefresherId,
  [in] long lFlags,
  [in] DWORD dwClientRefrVersion,
  [out] IWbemRemoteRefresher** ppRemRefresher,
  [out] GUID* pGuid,
  [out] DWORD* pdwSvrRefrVersion
);
```

pRefresherId: MUST be a pointer to the _WBEM_REFRESHER_ID structure, as specified in section 2.2.1.21, that identifies the client that is requesting refreshing services. This parameter MUST NOT be NULL.

lFlags: This parameter is not used, and it MUST be 0x0.

dwClientRefrVersion: MUST be the version of the client refresher. It MUST be 0x2.

ppRemRefresher: MUST be a pointer to a **IWbemRemoteRefresher** interface pointer that the client can use to call the **IWbemRemoteRefresher::RemoteRefresh** method to refresh CIM instances and enumerations. This parameter MUST NOT be NULL.

pGuid: MUST be an output parameter, which MUST be a pointer to a GUID value that MUST identify the returned refresher object. This parameter MUST NOT be NULL.

pdwSvrRefrVersion: MUST be an output parameter, which MUST be the version of the server refresher. It MUST be 0x1.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemRefreshingServices::GetRemoteRefresher** method MUST be called on the interface obtained using **IRemUnknown** and **IRemUnknown2** interfaces, as specified in [MS-DCOM], on the **IWbemServices** interface obtained in responses to a previous call to **IWbemLevel1Login::NTLMLogin** or **IWbemServices::OpenNamespace** method.

The **IWbemRefreshingServices::GetRemoteRefresher** method evaluates pRefresherID parameter and MUST return a **IWbemRemoteRefresher** interface pointer and a GUID randomly

generated by the server to identify this interface pointer. **IWbemRefreshingServices** interface pointer MUST have the same value as the one initially returned by **IWbemRefreshingServices::AddObjectToRefresher**, **IWbemRefreshingServices::AddObjectToRefresherByTemplate**, or **IWbemRefreshingServices::AddEnumToRefresher**.

A successful call to **IWbemRefreshingServices::GetRemoteRefresher** MUST return WBEM_S_NO_ERROR and fill the ppRemRefresher and pGuid. pdwSvrRefrVersion is reserved for future use and it MUST be set to 0x1.

The **IWbemRemoteRefresher** interface returned MUST be used in calls to **IWbemRemoteRefresher::RemoteRefresh** and **IWbemRemoteRefresher::StopRefreshing**.

3.1.5.12.6 IWbemRefreshingServices::ReconnectRemoteRefresher (Opnum 8)

The **IWbemRefreshingServices::ReconnectRemoteRefresher** method MUST restore a set of CIM instances and enumerations to a refresher by reconnecting to a server refresher after the client looses the connection to the server. The client SHOULD detect the loss of connection if it receives **RPC** disconnect errors, as specified in [MS-ERREF], from the server in response to **IWbemRemoteRefresher** operations.

```
HRESULT ReconnectRemoteRefresher(
  [in] _WBEM_REFRESHER_ID* pRefresherId,
  [in] long lFlags,
  [in] long lNumObjects,
  [in] DWORD dwClientRefrVersion,
  [in, size_is(lNumObjects)] _WBEM_RECONNECT_INFO* apReconnectInfo,
  [in, out, size_is(lNumObjects)]
    _WBEM_RECONNECT_RESULTS* apReconnectResults,
  [out] DWORD* pdwSvrRefrVersion
);
```

pRefresherId: MUST be a pointer to the _WBEM_REFRESHER_ID structure, as specified in section 2.2.1.21, identifying the client that is requesting refreshing services. This parameter MUST NOT be NULL.

lFlags: This parameter is not used, and it MUST be 0x0.

lNumObjects: MUST be the number of CIM instances contained in the apReconnectInfo array.

dwClientRefrVersion: MUST be the version of the client refresher. It MUST be 0x2.

apReconnectInfo: MUST be the pointer to the _WBEM_RECONNECT_INFO structure array specified in section 2.2.1.22 containing a type and a CIM path to the refresher objects. This parameter MUST NOT be NULL.

apReconnectResults: MUST be pointer to the _WBEM_RECONNECT_RESULTS structure array, which MUST contain the identifier for each CIM instance and enumeration, and the reconnection's success or failure status. This parameter MUST NOT be NULL.

pdwSvrRefrVersion: MUST be an output parameter that is the version of the server refresher. It MUST be 0x1.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

WBEM_S_NO_ERROR (0x00)

The **IWbemRefreshingServices::ReconnectRemoteRefresher** method MUST be called on the interface obtained using [MS-DCOM]**IRemUnknown** and **IRemUnknown2** interfaces, as specified in [MS-DCOM], on the **IWbemServices** interface previously obtained in responses to a previous call to **IWbemLevel1Login::NTLMLogin** or **IWbemServices::OpenNamespace** method.

The description of the IWbemRefreshingServices interface is specified in section **IWbemRefreshingServices Interface**.

In response to **IWbemRefreshingServices::ReconnectRemoteRefresher** the server MUST evaluate the pRefresherId and apReconnectInfo array and MUST reconnect to the refresher the requested CIM objects and enumerators listed in apReconnectInfo as specified in this section.

If one of the CIM objects cannot be reconnected the apReconnectResults element corresponding to apReconnectInfo MUST be set with an HRESULT return code.

A successful call into **IWbemRefreshingServices::ReconnectRemoteRefresher** MUST return WBEM_S_NO_ERROR and MUST fill the reconnection status into the apReconnectResults array.

The client MUST use the information provided in the apReconnectResults array. Each array element MUST contain a refresher CIM object identifier (m_lId member of _WBEM_RECONNECT_RESULTS). This identifier has the same meaning as the m_lCancelId from the _WBEM_REFRESH_INFO structure specified in section 2.2.1.20 and obtained from **IWbemRefreshingServices::AddObjectToRefresher**, **IWbemRefreshingServices::AddObjectToRefresherByTemplate**, or **IWbemRefreshingServices::AddEnumToRefresher**.

3.1.5.13 IWbemRemoteRefresher Interface

The **IWbemRemoteRefresher** interface (an [MS-DCOM] interface) MUST be implemented by the server and MUST be used to obtain updated information from the refreshing service. The interface MUST be uniquely identified by UUID F1E9C5B2-F59B-11d2-B362-00105A1F8177.<28>

Methods in RPC Opnum Order

Method	Description
RemoteRefresh	Retrieve the updated set of CIM instances and enumerations configured by a **IWbemRefreshingServices** interface pointer. Opnum: 3
StopRefreshing	Remove a set of CIM instance and enumerations configured by **IWbemRefreshingServices** interface pointer. Opnum: 4
GetGuid	This method is reserved for local use and is not used remotely. Opnum: 5

3.1.5.13.1 IWbemRemoteRefresher::RemoteRefresh (Opnum 3)

The **IWbemRemoteRefresher::RemoteRefresh** method MUST return the updated collection of CIM instances and enumerations previously configured by **IWbemRefreshingServices** interface pointer.

```
HRESULT RemoteRefresh(
  [in] long lFlags,
  [out] long* plNumObjects,
  [out, size_is(,*plNumObjects)]
    WBEM_REFRESHED_OBJECT** paObjects
);
```

lFlags: This parameter is not used, and it MUST be 0x0.

plNumObjects: If successful, *plNumObjects* MUST be a pointer to the number of CIM instances and enumerations that the method returns. It MUST NOT be NULL.

If the method fails, the server MUST set *plNumObjects* to NULL.

paObjects: If successful, *paObjects* MUST be a pointer to an array of WBEM_REFRESHED_OBJECT objects specified in section 2.2.1.16. The array MUST contain CIM instances and enumerations. It MUST NOT be NULL.

If the method fails, the server MUST set *paObjects* to NULL.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call.

The server MUST return WBEM_S_NO_ERROR (specified in section 2.2.1.11) to indicate the successful completion of the method.

If the method fails, the server MUST set the values referenced by **pINumObjects** and **paObjects** to NULL, and MUST return an error in the format specified in section 2.2.1.11.

WBEM_S_NO_ERROR (0x00)

The **IWbemRemoteRefresher::RemoteRefresh** method MUST be called on the **IWbemRemoteRefresher** interface pointer returned as a member of the _WBEM_REFRESH_INFO structure from **IWbemRefreshingServices** methods or on the interface returned by **IWbemRefreshingServices::GetRemoteRefresher** method invocation.

In response to **IWbemRemoteRefresher::RemoteRefresh** method, the server MUST read the current values of all the CIM objects previously added to the set of refreshing objects using **IWbemRefreshingServices** methods. The updated values for all CIM objects MUST be encoded into the output parameter using the format specified in this section.

3.1.5.13.2 IWbemRemoteRefresher::StopRefreshing (Opnum 4)

The **IWbemRemoteRefresher::StopRefreshing** method MUST remove a set of CIM instance or enumerations from the collection previously configured by **IWbemRefreshingServices** interface pointer.

```
HRESULT StopRefreshing(
  [in] long lNumIds,
  [in, size_is(lNumIds)] long* aplIds,
  [in] long lFlags
);
```

lNumIds: MUST be the number of identifiers in the array of object identifiers in the aplIds parameter.

aplIds: MUST be an array of object identifiers which MUST identify the CIM instances and enumerations to stop refreshing. The object identifier is the m_lCancelId member from the _WBEM_REFRESH_INFO structure specified in section 2.2.1.20 and MUST be obtained from a previous call to **IWbemRefreshingServices::AddObjectToRefresher**, **IWbemRefreshingServices::AddObjectToRefresherByTemplate**, or **IWbemRefreshingServices::AddEnumToRefresher** methods specified in section 3.1.5.12.

lFlags: This parameter is not used, and it MUST be 0x0.

Return Values: This method MUST return an HRESULT value that MUST indicate the status of the method call. In case of success, the server MUST return WBEM_S_NO_ERROR (as specified in section 2.2.1.11) to indicate the successful completion of the method. In case of failure the server MUST return an error in the format specified in section 2.2.1.11.

WBEM_S_NO_ERROR (0x00)

The **IWbemRemoteRefresher::StopRefreshing** method MUST be called on the **IWbemRemoteRefresher** interface pointer returned as a member of the _WBEM_REFRESH_INFO structure from **IWbemRefreshingServices** methods or on the interface returned by **IWbemRefreshingServices::GetRemoteRefresher** method invocation.

In response to **IWbemRemoteRefresher::StopRefreshing** method, the server MUST remove a list of CIM objects previously added to the set of refreshing objects using **IWbemRefreshingServices** methods. The CIM objects MUST be identified by their identifier, m_lCancelId member of _WBEM_REFRESH_INFO structure returned by previous **IWbemRefreshingServices::AddObjectToRefresher**, **IWbemRefreshingServices::AddObjectToRefresherByTemplate**, or **IWbemRefreshingServices::AddEnumToRefresher** calls.

In case of failure the server MUST return an error in the format specified in section 2.2.1.11.

3.1.5.13.3 IWbemRemoteRefresher::Opnum5NotUsedOnWire (Opnum 5)

The **IWbemRemoteRefresher::Opnum5NotUsedOnWire** method MUST return a random GUID identifying the server object receiving the call.

```
HRESULT Opnum5NotUsedOnWire(
  [in] long lFlags,
  [out] GUID* pGuid
);
```

lFlags: This parameter is not used, and it MUST be 0x0.

pGuid: MUST be an output parameter, which MUST be a pointer to a **GUID** value that MUST identify the server object. This parameter MUST NOT be NULL.<29>

3.1.6 Timer Events

The Windows Management Instrumentation Remote Protocol uses two timers:

- Sink timer: If the timer expires and the call is not completed, the server MUST cancel the asynchronous operation for which the timer expired.
- Backup Timer: If the timer expires, the server MUST resume operations by simulating **IWbemBackupRestoreEx::Resume** and MUST reset the timer to 0.

3.1.7 Other Local Events

None.

4 Protocol Examples

The following sections describe several operations as used in common scenarios to illustrate the function of the Windows Management Instrumentation Remote Protocol.

4.1 Protocol Initialization

The protocol MUST be initiated by the DCOM Remote Protocol activation of CLSID _IWbemLevel1Login.

For a client application to connect to the WMI service on a remote server, the client application MUST first obtain an **IWbemLevel1Login** interface pointer to the server on the remote computer by using the COM activation. The client MUST use the DCOM Remote Protocol to obtain an **IWbemLevel1Login** interface pointer.

The client MUST call **IWbemLevel1Login::NTLMLogin** method to obtain an **IWbemServices**. The **IWbemServices** interface pointer is the starting point for all other activities.

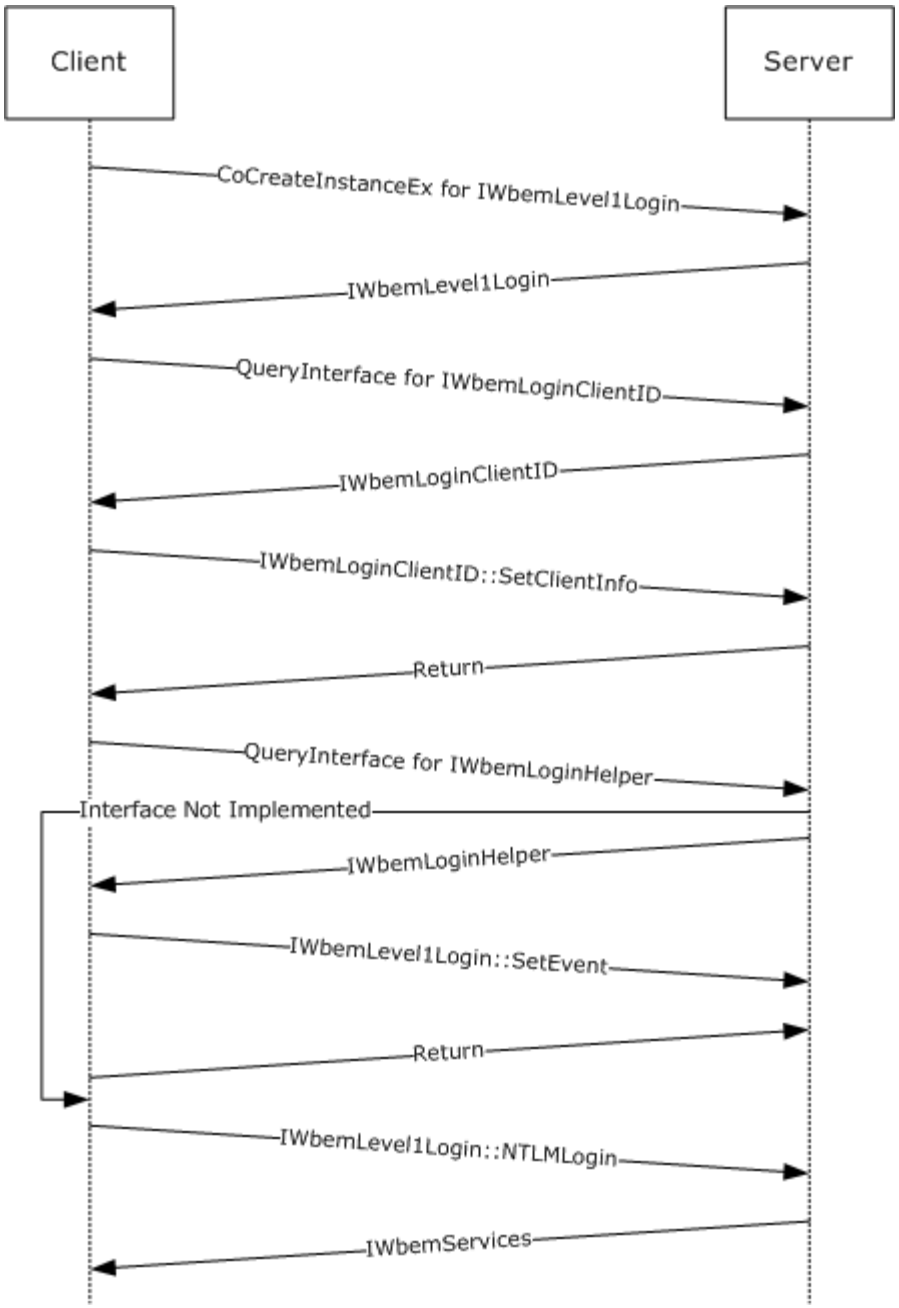


Figure 8: Protocol initialization example

4.2 Synchronous Operations

A synchronous operation completes when the entire result set is ready on the server. The following sections describe the different scenarios where synchronous operations are applicable.

4.2.1 Synchronous Delivery of a Single Result

The **IWbemServices::GetObject** and **IWbemServices::ExecMethod** methods support synchronous calls returning a single marshaled IWbemClassObject interface pointer.

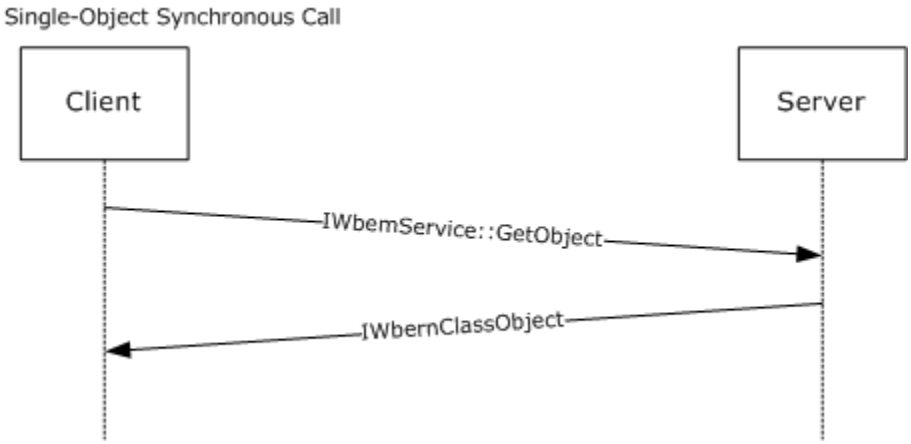


Figure 9: Synchronous delivery of a single result

4.2.2 Synchronous Delivery of Result Sets

In this context of operation, four cases MUST be distinguished, depending on the server and client version:

- Windows 2000 client and prior with Windows 2000 Server and prior.
- Windows 2000 client and prior with post-Windows 2000 Server.
- Post-Windows 2000 client with post-Windows 2000 Server.
- Post-Windows 2000 client with Windows 2000 Server and prior.

4.2.2.1 Windows 2000 Client and Prior with Windows 2000 Server and Prior

To make a synchronous operation from a client to a server, the client uses the **IWbemServices** interface pointer. The client calls the **IWbemServices** synchronous methods **IWbemServices::CreateInstanceEnum**, **IWbemServices::CreateClassEnum**, and **IWbemServices::ExecQuery**. In response to the method executed, the server returns an **IEnumWbemClassObject** interface pointer. The client then uses the **IEnumWbemClassObject::Next** method to repeatedly retrieve the IWbemClassObject objects from the query result set.

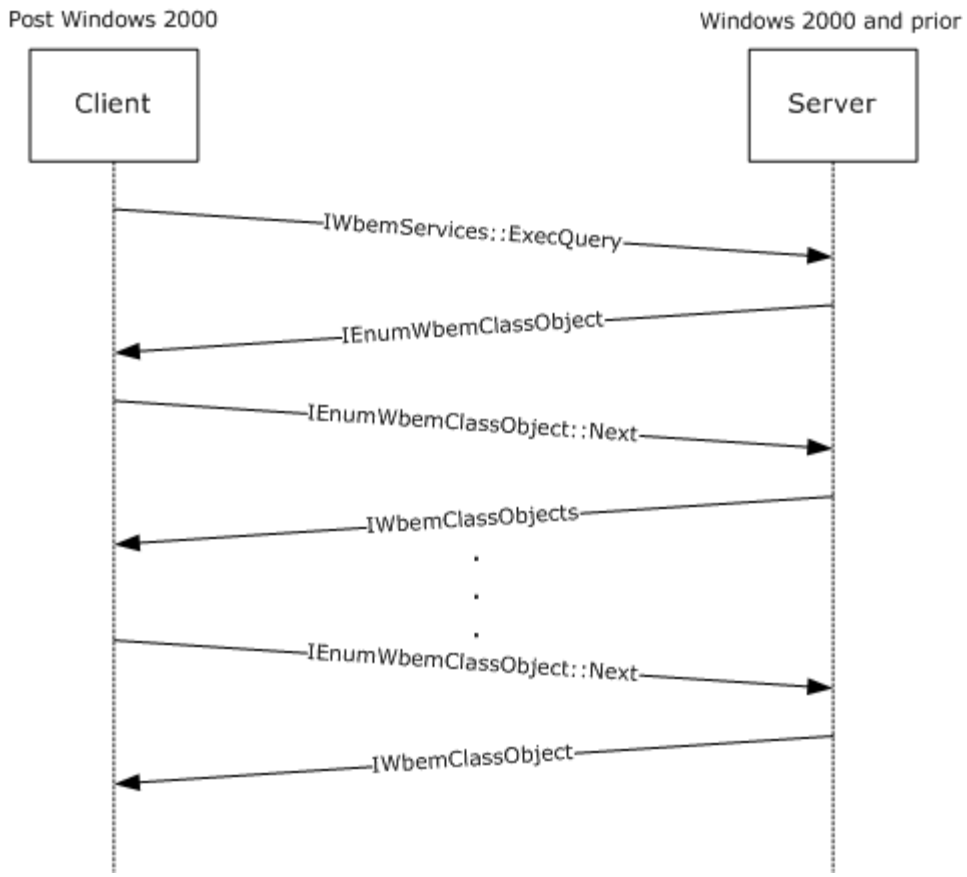


Figure 10: Windows 2000 client and prior with Windows 2000 server and prior

4.2.2.2 Windows 2000 Client and Prior with Post-Windows 2000 Server

To make a synchronous operation from a client to a server, the client uses the **IWbemServices** interface pointer. The client calls the **IWbemServices** synchronous methods **IWbemServices::CreateInstanceEnum**, **IWbemServices::CreateClassEnum**, and **IWbemServices::ExecQuery**. In response to the method executed, the server returns an **IEnumWbemClassObject** interface pointer. The client then uses the **IEnumWbemClassObject::Next** method to repeatedly retrieve the IWbemClassObject objects from the query result set.

The call sequence here is the same as that in 4.2.2.1 because in both cases, the client is "Windows 2000 and Prior". Therefore the client still uses the old mechanism for communication. Further sections 4.2.2.3 and 4.2.2.4 explain the call sequences between the newer versions of client and server.

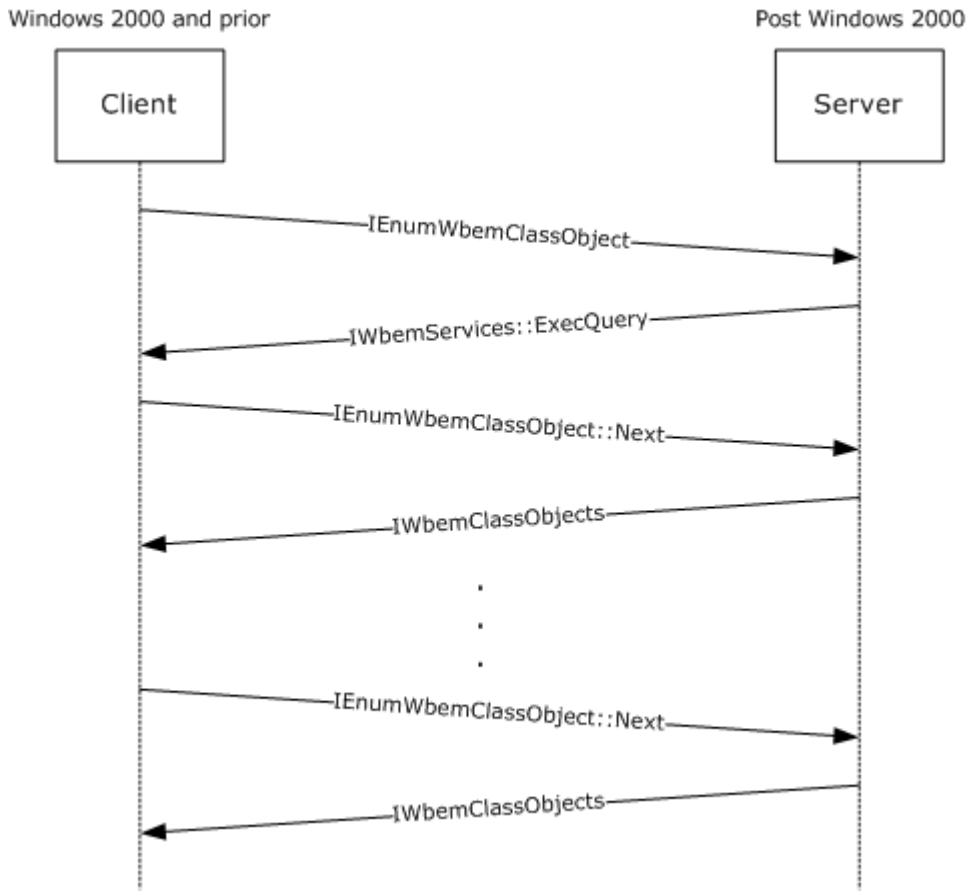


Figure 11: Windows 2000 client and prior with post-Windows 2000 server

4.2.2.3 Post-Windows 2000 Client with Post-Windows 2000 Server

To make a synchronous operation from a client to a server, the client uses the **IWbemServices** interface pointer. The client calls the **IWbemServices** synchronous methods **IWbemServices::CreateInstanceEnum**, **IWbemServices::CreateClassEnum**, and **IWbemServices::ExecQuery**. In response to the method executed, the server returns an **IEnumWbemClassObject** interface pointer. The client uses **IRemUnknown** and **IRemUnknown2**, as specified in [MS-DCOM], on to obtain an **IWbemFetchSmartEnum** interface pointer from **IEnumWbemClassObject** interface pointer. The client then calls the **IWbemFetchSmartEnum::GetSmartEnum** method to obtain the **IWbemWCOSmartEnum** interface pointer. The client uses the **IWbemWCOSmartEnum::Next** method repeatedly to retrieve the IWbemClassObject interface pointers that contain the result set. The results are encoded as an ObjectArray as specified in section 2.2.1.15.

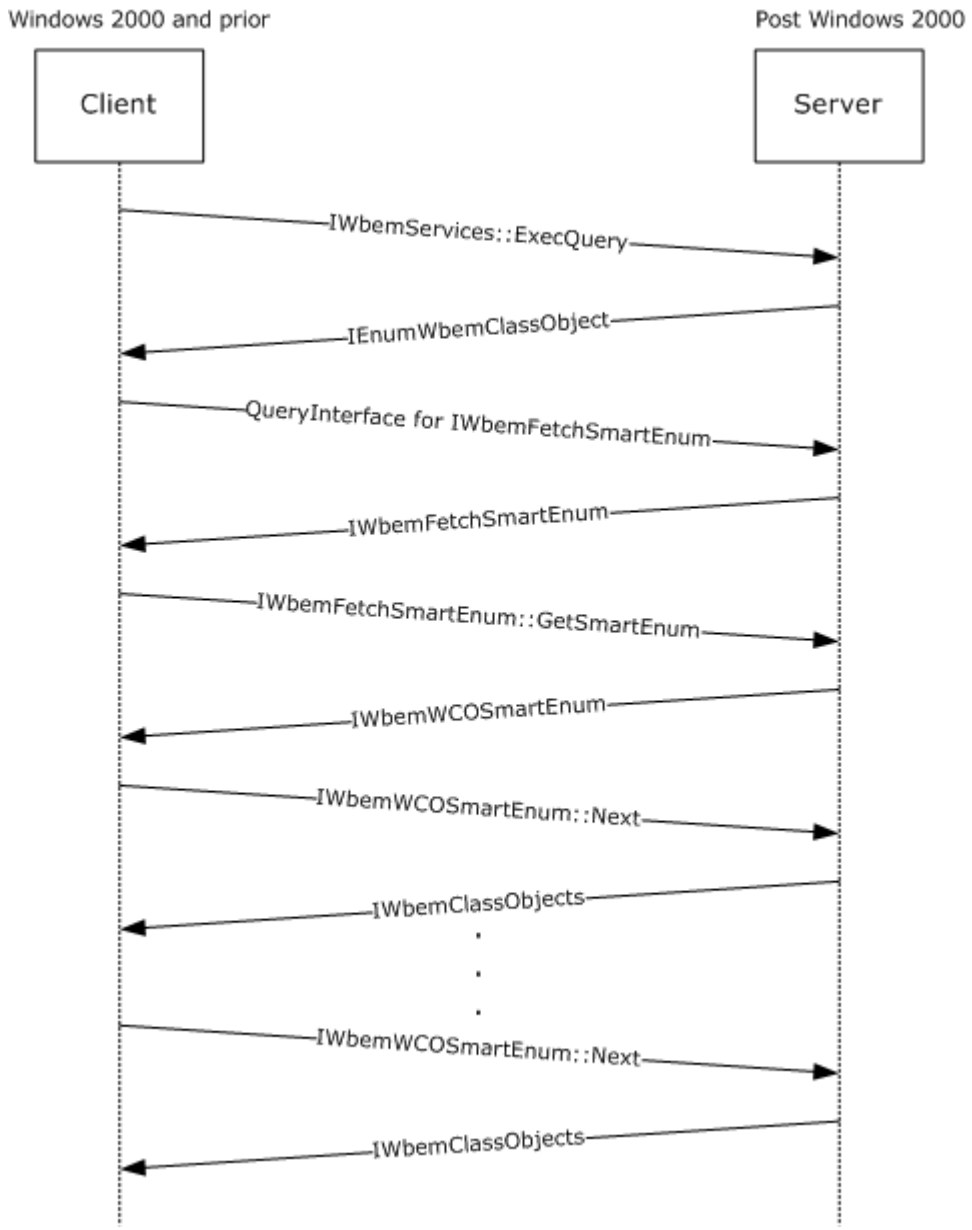


Figure 12: Post-Windows 2000 client with post-Windows 2000 server

4.2.2.4 Post-Windows 2000 Client with Windows 2000 Server and Prior

To make a synchronous operation from a client to a server, the client uses the **IWbemServices** interface pointer. The client calls the **IWbemServices** synchronous methods **IWbemServices::CreateInstanceEnum**, **IWbemServices::CreateClassEnum**, and **IWbemServices::ExecQuery**. In response to the method executed, the server returns an **IEnumWbemClassObject** interface pointer. The client uses **IRemUnknown** and **IRemUnknown2**, as specified in [MS-DCOM], on to obtain an **IWbemFetchSmartEnum** interface pointer from **IEnumWbemClassObject** interface pointer. The operation fails because the server is

not implementing the **IWbemFetchSmartEnum** interface. The client falls back to the Windows 2000 client and earlier behavior.

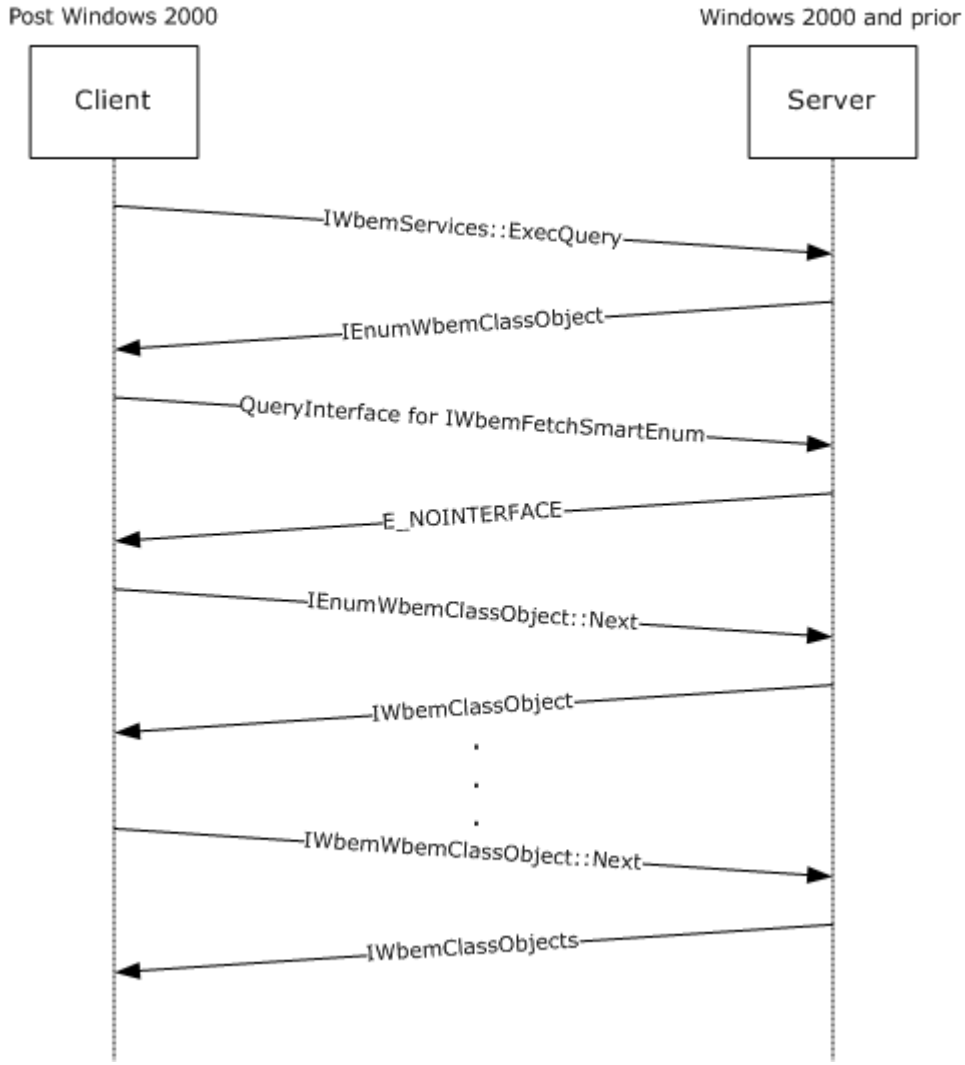


Figure 13: Post-Windows 2000 client with Windows 2000 server and prior

4.3 Semisynchronous Operations

In semisynchronous cases, the call MUST return immediately, and another interface MUST be used to retrieve the operation results. The returned interface depends on the **IWbemServices** methods invoked by the client. The following sections describe the two different behaviors.

4.3.1 Semisynchronous Delivery of a Single Result

The methods returning a single element such as **IWbemServices::OpenNamespace**, **IWbemServices::GetObject**, **IWbemServices::PutClass**, **IWbemServices::DeleteClass**, **IWbemServices::PutInstance**, **IWbemServices::DeleteInstance**, and **IWbemServices::ExecMethod** return an **IWbemCallResult** interface pointer. To obtain the operation results, the client MUST use the **IWbemCallResult** methods.

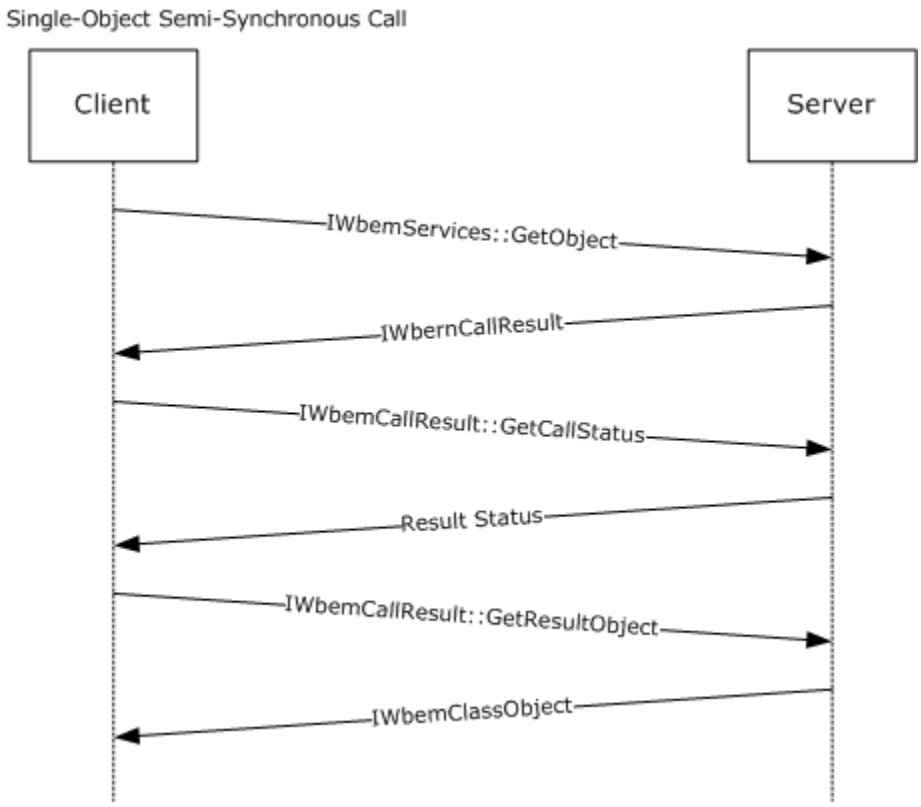


Figure 14: Semisynchronous delivery of a single result

4.3.2 Semisynchronous Delivery of Result Sets

The semisynchronous operation uses the same sequence as the synchronous calls, as specified in section 4.2.2, to request a result set. However, in semisynchronous cases, the **IEnumWbemClassObject** interface pointer MUST be returned before the result set is available on the server. This is different from the synchronous case, in which the interface pointer is returned only after the result set is available on the server. The **IEnumWbemClassObject** interface pointer MUST be returned before the result set is available on the server. When the client calls the **IEnumWbemClassObject::Next** method, it MUST specify a timeout within the server MUST return the requested results. When one of the previous conditions is satisfied, the call completes.

4.4 Asynchronous Delivery of Results

Asynchronous operations complete immediately on the client, while the server continues to execute the request. A response handler MUST be used to receive the results as they become available. The **IWbemObjectSink::Indicate** method MUST be called by the server when a result is found during the operation.

To make an asynchronous query, the client MUST use the **IWbemServices** interface pointer and passes the **IWbemObjectSink** interface when calling an asynchronous method of the **IWbemServices** interface.

If the asynchronous call returns SUCCESS, the server MUST keep a reference to the **IWbemObjectSink** interface pointer. The server MUST deliver the results through the **IWbemObjectSink::Indicate** method, if the server is required to return WMI objects to the client.

After the delivery of all objects, the server MUST make a single call to **IWbemObjectSink::SetStatus** to indicate that the operation finished, containing the final status of the operation. After that, the server MUST release the **IWbemObjectSink** pointer.

In case the asynchronous call does not return SUCCESS, the server MUST NOT use the **IWbemObjectSink** pointer and the server MUST NOT keep reference to the **IWbemObjectSink**.

Figure 15: Asynchronous delivery of results

4.5 Optimized Asynchronous Delivery of Results

The asynchronous communication is optimized<30> to reduce the network usage by making use of the ObjectArray structure as specified in section 2.2.1.15.

A client supporting that capability MUST notify the server by returning 0x400FF (WBEM_S_NEW_STYLE) in the first Indicate operation. A server not supporting the ObjectArray structure MUST interpret this as a success return code, while a server supporting the ObjectArray structure MUST notice the code and MUST send the rest of the results using the ObjectArray structure, as specified in section 2.2.1.15.

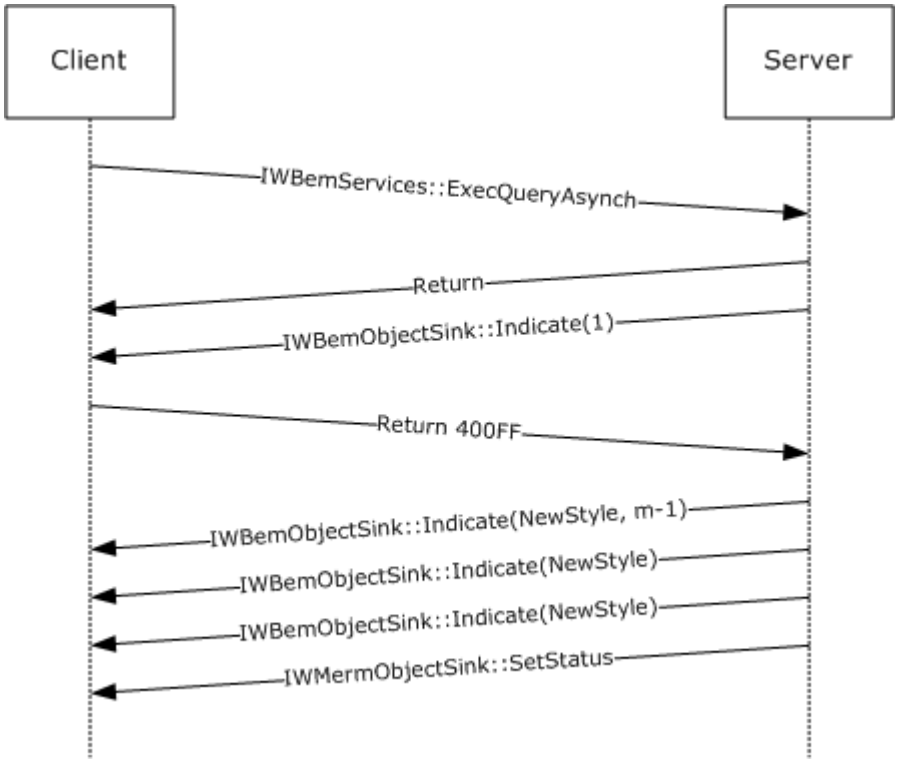


Figure 16: Optimized asynchronous delivery of results

4.6 Configuring Refreshing Services

When using the refresher mechanism, a client application that is connected to a remote computer through an **IWbemServices** pointer MUST use **IRemUnknown** and **IRemUnknown2**, as specified in [MS-DCOM], to obtain an **IWbemRefreshingServices** interface, and it MUST add CIM objects or enumerators as needed. The following diagram illustrates this behavior.

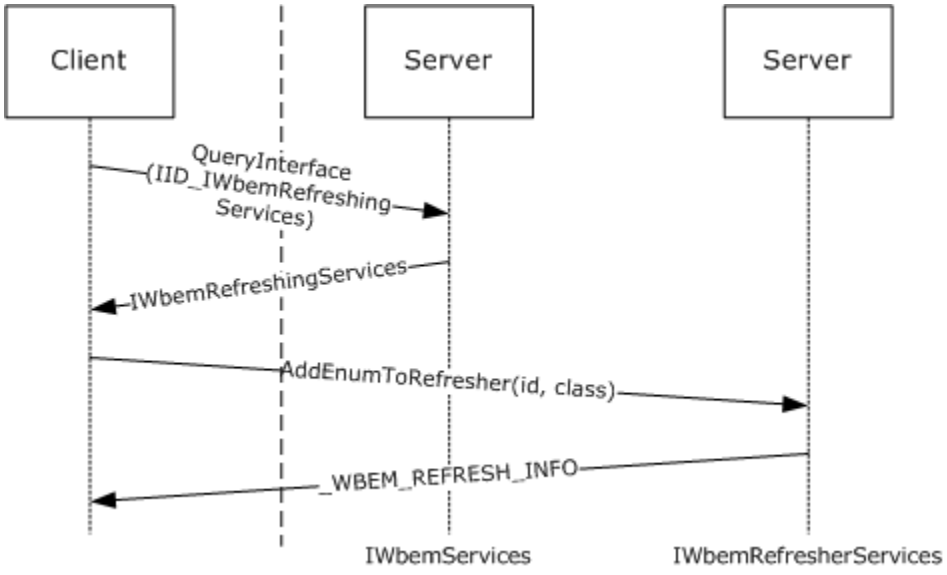


Figure 17: Configuring refreshing services

4.7 Using the Refresher Interface

The **IWbemRemoteRefresher** interface pointer returned from **IWbemRefreshingServices** is used to obtain an updated result set. For the usage of the remote refresher, the client MUST call the **IWbemRemoteRefresher::RemoteRefresh** method any time the client needs to get an updated set of data.

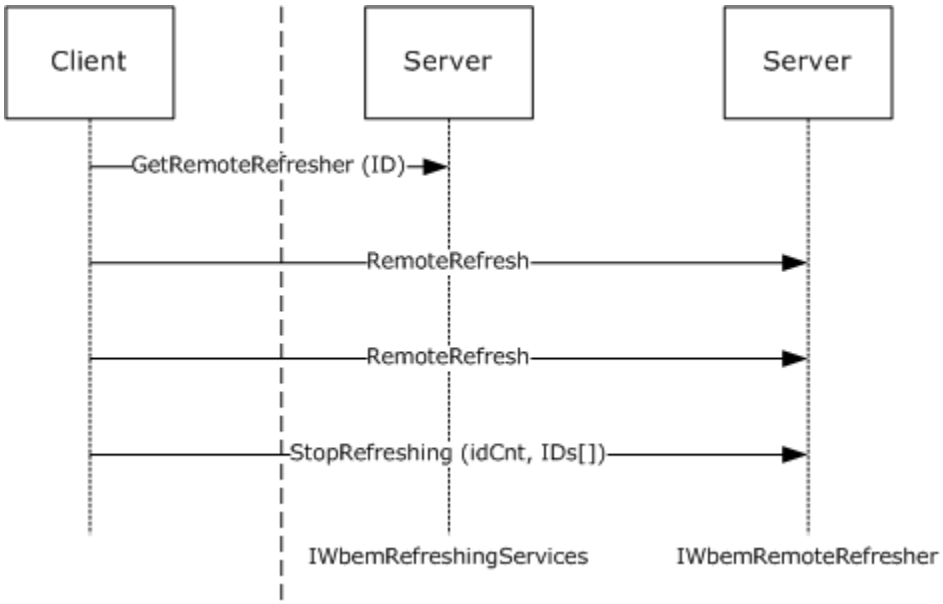


Figure 18: Using the refresher interface

5 Security

The following sections specify security considerations for implementers of the Windows Management Instrumentation Remote Protocol.

5.1 Security Considerations for Implementers

For all methods, the server MUST evaluate the authentication level and the security principal rights to open a CIM namespace, and fail the operation if the security requirements are not met.<31>

5.2 Index of Security Parameters

The server MUST secure the access to each CIM namespace using standard Windows security descriptors<32>, as specified in [MS-DTYP]. The access mask controlling the security principal rights contains the following specific rights interpreted as specified in the table below.

Constants	Value	Meaning
WBEM_ENABLE	0x1	Grants the security principal read permissions.
WBEM_FULL_WRITE_REP	0x4	Grants the security principal to write to classes and instances.
WBEM_METHOD_EXECUTE	0x2	Grants the security principal to execute methods.
WBEM_PARTIAL_WRITE_REP	0x8	Grants the security principal to update information in the cache only.
WBEM_REMOTE_ACCESS	0x20	Grants the security principal to remotely access the server.
WBEM_WRITE_PROVIDER	0x10	Grants the security principal to update real-time information only.

The namespace security descriptor MAY be changed using the Windows Management Instrumentation Remote Protocol and the required CIM object encoding, as specified in [MS-WMIO]. To query or change the security descriptor, the __SystemSecurity class methods GetSD and SetSD ([MSDN-WMISYSCLASSES]) MUST be used. The __SystemSecurity class MUST be implemented at the top level of every namespace to manage the namespace security. The GetSD and SetSD methods are invoked as specified in sections 3.1.5.2.22 and 3.1.5.2.23.

6 Appendix A: Full IDL

For ease of implementation, the full IDL is provided, where "ms-oaut.idl" is the IDL found in [MS-OAUT] appendix A.

```
import "ms-dtyp.idl";
import "ms-oaut.idl";

typedef GUID *REFGUID;

interface IWbemClassObject;
interface IWbemServices;
interface IWbemObjectSink;
interface IEnumWbemClassObject;
interface IWbemCallResult;
interface IWbemContext;
interface IWbemBackupRestore;
interface IWbemBackupRestoreEx;
interface IWbemLoginClientID;
interface IWbemLevel1Login;
interface IWbemLoginHelper;

[
    restricted,
    uuid(8BC3F05E-D86B-11d0-A075-00C04FB68820)
]
coclass WbemLevel1Login {
    interface IWbemLevel1Login;
};

typedef long HRESULT;

typedef [v1_enum] enum tag_WBEM_REFRESHER_FLAGS {
    WBEM_FLAG_REFRESH_AUTO_RECONNECT = 0,
    WBEM_FLAG_REFRESH_NO_AUTO_RECONNECT = 1
} WBEM_REFRESHER_FLAGS;

typedef [v1_enum] enum tag_WBEM_QUERY_FLAG_TYPE {
    WBEM_FLAG_DEEP = 0,
    WBEM_FLAG_SHALLOW = 1,
    WBEM_FLAG_PROTOTYPE = 2
} WBEM_QUERY_FLAG_TYPE;


typedef [v1_enum] enum tag_WBEM_CHANGE_FLAG_TYPE {
    WBEM_FLAG_UPDATE_ONLY = 0x01,
    WBEM_FLAG_CREATE_ONLY = 0x02,
    WBEM_FLAG_UPDATE_SAFE_MODE = 0x20,
    WBEM_FLAG_UPDATE_FORCE_MODE = 0x40,
} WBEM_CHANGE_FLAG_TYPE;

typedef [v1_enum] enum tag_WBEM_CONNECT_OPTIONS {
    WBEM_FLAG_CONNECT_REPOSITORY_ONLY =  0X40,
    WBEM_FLAG_CONNECT_PROVIDERS       = 0X100
} WBEM_CONNECT_OPTIONS;
```

```
typedef [v1_enum] enum tag_WBEM_GENERIC_FLAG_TYPE {
    WBEM_FLAG_RETURN_IMMEDIATELY = 0x10,
    WBEM_FLAG_FORWARD_ONLY = 0x20,
    WBEM_FLAG_SEND_STATUS = 0x80,
    WBEM_FLAG_ENSURE_LOCATABLE = 0x100,
    WBEM_FLAG_DIRECT_READ = 0x200,
    WBEM_FLAG_USE_AMENDED_QUALIFIERS,
} WBEM_GENERIC_FLAG_TYPE;

typedef enum tag_WBEM_STATUS_TYPE {
    WBEM_STATUS_COMPLETE = 0,
    WBEM_STATUS_PROGRESS = 2
} WBEM_STATUS_TYPE;

typedef [v1_enum] enum tag_WBEM_TIMEOUT_TYPE {
    WBEM_NO_WAIT = 0,
    WBEM_INFINITE = 0xFFFFFFFF
} WBEM_TIMEOUT_TYPE;

typedef [v1_enum] enum tag_WBEM_BACKUP_RESTORE_FLAGS {
    WBEM_FLAG_BACKUP_RESTORE_FORCE_SHUTDOWN = 1
} WBEM_BACKUP_RESTORE_FLAGS;


typedef [v1_enum] enum tag_WBEMSTATUS {
    WBEM_S_NO_ERROR = 0x00,
    WBEM_S_FALSE = 0x01,
    WBEM_S_TIMEDOUT = 0x40004,
    WBEM_E_FAILED = 0x80041001,
    WBEM_E_NOT_AVAILABLE = 0x80041009,
    WBEM_E_NOT_SUPPORTED = 0x8004100C,
    WBEM_E_INVALID_OPERATION = 0x80041016,
    E_NOTIMPL = 0x80004001,
}   WBEMSTATUS;


[
    restricted,
    uuid(674B6698-EE92-11d0-AD71-00C04FD8FDFF)
]
coclass WbemContext
{
    interface IWbemContext;
};


[
    uuid(9A653086-174F-11d2-B5F9-00104B703EFD)
]
coclass WbemClassObject
{
    interface IWbemClassObject;
};


[
    uuid(C49E32C6-BC8B-11d2-85D4-00105A1F8304)
```

```
]
coclass WbemBackupRestore
{
    interface IWbemBackupRestoreEx;
};

#define OPTIONAL in, unique

interface IWbemQualifierSet;

[
    local,
    restricted,
    object,
    uuid(dc12a681-737f-11cf-884d-00aa004b2e24)
]
interface IWbemClassObject : IUnknown
{
};

interface IWbemServices;


[
    object,
    restricted,
    uuid(7c857801-7381-11cf-884d-00aa004b2e24)
]
interface IWbemObjectSink : IUnknown
{
    HRESULT SetStatus(
        [in] long lFlags,
        [in] HRESULT hResult,
        [in] BSTR strParam,
        [in] IWbemClassObject* pObjParam
    );

    HRESULT Indicate(
        [in] long lObjectCount,
        [in, size_is(lObjectCount)]
            IWbemClassObject** apObjArray
    );
};

[
    object,
    restricted,
    uuid(027947e1-d731-11ce-a357-000000000001)
]
interface IEnumWbemClassObject : IUnknown
{
    HRESULT Reset();

    HRESULT Next(
        [in]  long lTimeout,
        [in]  ULONG uCount,
        [out, size_is(uCount), length_is(*puReturned)]
              IWbemClassObject** apObjects,
```

```
        [out] ULONG* puReturned
    );

    HRESULT NextAsync(
        [in]  ULONG uCount,
        [in]  IWbemObjectSink* pSink
    );

    HRESULT Clone(
        [out] IEnumWbemClassObject** ppEnum
    );

    HRESULT Skip(
        [in] long lTimeout,
        [in] ULONG nCount
    );
};


[
    object,
    restricted,
    local,
    uuid(44aca674-e8fc-11d0-a07c-00c04fb68820)
]
interface IWbemContext : IUnknown
{
};


[
    object,
    restricted,
    uuid(44aca675-e8fc-11d0-a07c-00c04fb68820)
]
interface IWbemCallResult : IUnknown
{
    HRESULT GetResultObject(
        [in]  long lTimeout,
        [out] IWbemClassObject** ppResultObject
    );

    HRESULT GetResultString(
        [in]  long lTimeout,
        [out] BSTR* pstrResultString
    );

    HRESULT GetResultServices(
        [in]  long lTimeout,
        [out] IWbemServices** ppServices
    );

    HRESULT GetCallStatus(
        [in]  long lTimeout,
        [out] long* plStatus
    );
};
```

```
[
    object,
    restricted,
    uuid(9556dc99-828c-11cf-a37e-00aa003240c7),
    pointer_default(unique)
]
interface IWbemServices : IUnknown
{
    HRESULT OpenNamespace(
        [in]  const BSTR strNamespace,
        [in]  long lFlags,
        [in]  IWbemContext* pCtx,
        [out, in, unique] IWbemServices** ppWorkingNamespace,
        [out, in, unique] IWbemCallResult** ppResult
    );

    HRESULT CancelAsyncCall(
        [in] IWbemObjectSink* pSink
    );

    HRESULT QueryObjectSink(
        [in] long lFlags,
        [out] IWbemObjectSink** ppResponseHandler
    );

    HRESULT GetObject(
        [in]  const BSTR strObjectPath,
        [in]  long lFlags,
        [in]  IWbemContext* pCtx,
        [out, in, unique] IWbemClassObject** ppObject,
        [out, in, unique] IWbemCallResult** ppCallResult
    );

    HRESULT GetObjectAsync(
        [in]  const BSTR strObjectPath,
        [in]  long lFlags,
        [in]  IWbemContext* pCtx,
        [in]  IWbemObjectSink* pResponseHandler
    );

    HRESULT PutClass(
        [in] IWbemClassObject* pObject,
        [in] long lFlags,
        [in]  IWbemContext* pCtx,
        [out, in, unique] IWbemCallResult** ppCallResult
    );

    HRESULT PutClassAsync(
        [in] IWbemClassObject* pObject,
        [in] long lFlags,
        [in] IWbemContext* pCtx,
        [in] IWbemObjectSink* pResponseHandler
    );

    HRESULT DeleteClass(
        [in] const BSTR strClass,
        [in] long lFlags,
```

```
    [in] IWbemContext* pCtx,
    [out, in, unique] IWbemCallResult** ppCallResult
);

HRESULT DeleteClassAsync(
    [in] const BSTR strClass,
    [in] long lFlags,
    [in] IWbemContext* pCtx,
    [in] IWbemObjectSink* pResponseHandler
);

HRESULT CreateClassEnum(
    [in]  const BSTR strSuperclass,
    [in]  long lFlags,
    [in]  IWbemContext* pCtx,
    [out] IEnumWbemClassObject** ppEnum
);

HRESULT CreateClassEnumAsync(
    [in]  const BSTR strSuperclass,
    [in]  long lFlags,
    [in]  IWbemContext* pCtx,
    [in]  IWbemObjectSink* pResponseHandler
);

HRESULT PutInstance(
    [in]  IWbemClassObject* pInst,
    [in]  long lFlags,
    [in]  IWbemContext* pCtx,
    [out, in, unique] IWbemCallResult** ppCallResult
);

HRESULT PutInstanceAsync(
    [in] IWbemClassObject* pInst,
    [in] long lFlags,
    [in] IWbemContext* pCtx,
    [in] IWbemObjectSink* pResponseHandler
);

HRESULT DeleteInstance(
    [in] const BSTR strObjectPath,
    [in] long lFlags,
    [in]  IWbemContext* pCtx,
    [out, in, unique] IWbemCallResult** ppCallResult
);

HRESULT DeleteInstanceAsync(
    [in] const BSTR strObjectPath,
    [in] long lFlags,
    [in] IWbemContext* pCtx,
    [in] IWbemObjectSink* pResponseHandler
);

HRESULT CreateInstanceEnum(
    [in] const BSTR strFilter,
    [in] long lFlags,
    [in]  IWbemContext* pCtx,
    [out] IEnumWbemClassObject** ppEnum
```

```
);

HRESULT CreateInstanceEnumAsync(
    [in] const BSTR strSuperClass,
    [in] long lFlags,
    [in] IWbemContext* pCtx,
    [in] IWbemObjectSink* pResponseHandler
);

HRESULT ExecQuery(
    [in]  const BSTR strQueryLanguage,
    [in]  const BSTR strQuery,
    [in]  long lFlags,
    [in]  IWbemContext* pCtx,
    [out] IEnumWbemClassObject** ppEnum
);

HRESULT ExecQueryAsync(
    [in] const BSTR strQueryLanguage,
    [in] const BSTR strQuery,
    [in] long lFlags,
    [in] IWbemContext* pCtx,
    [in] IWbemObjectSink* pResponseHandler
);

HRESULT ExecNotificationQuery(
    [in] const BSTR strQueryLanguage,
    [in] const BSTR strQuery,
    [in] long lFlags,
    [in]  IWbemContext* pCtx,
    [out] IEnumWbemClassObject** ppEnum
);

HRESULT ExecNotificationQueryAsync(
    [in] const BSTR strQueryLanguage,
    [in] const BSTR strQuery,
    [in] long lFlags,
    [in] IWbemContext* pCtx,
    [in] IWbemObjectSink* pResponseHandler
);

HRESULT ExecMethod(
    [in] const BSTR strObjectPath,
    [in] const BSTR strMethodName,
    [in] long lFlags,
    [in] IWbemContext* pCtx,
    [in] IWbemClassObject* pInParams,
    [out, in, unique] IWbemClassObject** ppOutParams,
    [out, in, unique] IWbemCallResult** ppCallResult
);

HRESULT ExecMethodAsync(
    [in] const BSTR strObjectPath,
    [in] const BSTR strMethodName,
    [in] long lFlags,
    [in] IWbemContext* pCtx,
    [in] IWbemClassObject* pInParams,
    [in] IWbemObjectSink* pResponseHandler
```

```
    );
};

[
    object,
    restricted,
    uuid(C49E32C7-BC8B-11d2-85D4-00105A1F8304)
]
interface IWbemBackupRestore : IUnknown
{
    HRESULT Backup(
        [in, string] LPCWSTR strBackupToFile,
        [in] long lFlags
    );

    HRESULT Restore(
        [in, string] LPCWSTR strRestoreFromFile,
        [in] long lFlags
    );
};


[
    object,
    restricted,
    uuid(A359DEC5-E813-4834-8A2A-BA7F1D777D76)
]
interface IWbemBackupRestoreEx : IWbemBackupRestore
{
    HRESULT Pause();
    HRESULT Resume();
};


typedef enum _WBEM_REFR_VERSION_NUMBER {
    WBEM_REFRESHER_VERSION  = 2
} WBEM_REFR_VERSION_NUMBER;

typedef enum _WBEM_INSTANCE_BLOB_TYPE {
    WBEM_BLOB_TYPE_ALL = 2,
    WBEM_BLOB_TYPE_ERROR = 3,
    WBEM_BLOB_TYPE_ENUM = 4
} WBEM_INSTANCE_BLOB_TYPE;

typedef struct _WBEM_REFRESHED_OBJECT {
    long m_lRequestId;
    WBEM_INSTANCE_BLOB_TYPE m_lBlobType;
    long m_lBlobLength;
    [size_is(m_lBlobLength)] byte* m_pbBlob;
} WBEM_REFRESHED_OBJECT;

[
    restricted,
    uuid(F1E9C5B2-F59B-11d2-B362-00105A1F8177)
]
interface IWbemRemoteRefresher : IUnknown {
    HRESULT RemoteRefresh(
        [in] long lFlags,
```

```
        [out] long* plNumObjects,
        [out, size_is(,*plNumObjects)]
            WBEM_REFRESHED_OBJECT** paObjects
    );

    HRESULT StopRefreshing(
        [in] long lNumIds,
        [in, size_is(lNumIds)] long* aplIds,
        [in] long lFlags
    );

    HRESULT Opnum5NotUsedOnWire(
        [in] long lFlags,
        [out] GUID* pGuid
    );
};


typedef struct {
    IWbemRemoteRefresher* m_pRefresher;
    IWbemClassObject*     m_pTemplate;
    GUID                  m_Guid;
} _WBEM_REFRESH_INFO_REMOTE;

typedef struct {
    [string] wchar_t* m_wszNamespace;
    IWbemClassObject* m_pTemplate;
} _WBEM_REFRESH_INFO_NON_HIPERF;

typedef enum
{
    WBEM_REFRESH_TYPE_INVALID = 0,
    WBEM_REFRESH_TYPE_REMOTE = 3,
    WBEM_REFRESH_TYPE_NON_HIPERF = 6
}WBEM_REFRESH_TYPE;

typedef [switch_type(long)] union {
    [case (WBEM_REFRESH_TYPE_REMOTE)]
        _WBEM_REFRESH_INFO_REMOTE m_Remote;

    [case (WBEM_REFRESH_TYPE_NON_HIPERF)]
        _WBEM_REFRESH_INFO_NON_HIPERF m_NonHiPerf;

    [case (WBEM_REFRESH_TYPE_INVALID)]
        HRESULT m_hres;

} WBEM_REFRESH_INFO_UNION;

typedef struct {
    long m_lType;
    [switch_is(m_lType)] WBEM_REFRESH_INFO_UNION m_Info;
    long m_lCancelId;
} _WBEM_REFRESH_INFO;

typedef struct {
    [string] LPSTR m_szMachineName;
    DWORD          m_dwProcessId;
    GUID           m_guidRefresherId;
```

```
} _WBEM_REFRESHER_ID;

typedef enum {
   WBEM_RECONNECT_TYPE_OBJECT = 0,
   WBEM_RECONNECT_TYPE_ENUM = 1,
   WBEM_RECONNECT_TYPE_LAST = 2
}WBEM_RECONNECT_TYPE;

typedef struct {
    long m_lType;
    [string] LPCWSTR m_pwcsPath;
} _WBEM_RECONNECT_INFO;

typedef struct {
   long m_lId;
   HRESULT m_hr;
} _WBEM_RECONNECT_RESULTS;

[
    restricted,
    uuid(2C9273E0-1DC3-11d3-B364-00105A1F8177)
]
interface IWbemRefreshingServices : IUnknown
{
    HRESULT AddObjectToRefresher(
        [in] _WBEM_REFRESHER_ID* pRefresherId,
        [in, string] LPCWSTR wszPath,
        [in] long lFlags,
        [in] IWbemContext* pContext,
        [in] DWORD dwClientRefrVersion,
        [out] _WBEM_REFRESH_INFO* pInfo,
        [out] DWORD* pdwSvrRefrVersion
    );

    HRESULT AddObjectToRefresherByTemplate(
        [in] _WBEM_REFRESHER_ID* pRefresherId,
        [in] IWbemClassObject* pTemplate,
        [in] long lFlags,
        [in] IWbemContext* pContext,
        [in] DWORD dwClientRefrVersion,
        [out] _WBEM_REFRESH_INFO* pInfo,
        [out] DWORD* pdwSvrRefrVersion
    );

    HRESULT AddEnumToRefresher(
        [in] _WBEM_REFRESHER_ID* pRefresherId,
        [in, string] LPCWSTR wszClass,
        [in] long lFlags,
        [in] IWbemContext* pContext,
        [in] DWORD dwClientRefrVersion,
        [out] _WBEM_REFRESH_INFO* pInfo,
        [out] DWORD* pdwSvrRefrVersion
    );

    HRESULT RemoveObjectFromRefresher(
        [in] _WBEM_REFRESHER_ID* pRefresherId,
        [in] long lId,
        [in] long lFlags,
```

```
        [in] DWORD dwClientRefrVersion,
        [out] DWORD* pdwSvrRefrVersion
    );

    HRESULT GetRemoteRefresher(
        [in] _WBEM_REFRESHER_ID* pRefresherId,
        [in] long lFlags,
        [in] DWORD dwClientRefrVersion,
        [out] IWbemRemoteRefresher** ppRemRefresher,
        [out] GUID* pGuid,
        [out] DWORD* pdwSvrRefrVersion
    );

    HRESULT ReconnectRemoteRefresher(
        [in] _WBEM_REFRESHER_ID* pRefresherId,
        [in] long lFlags,
        [in] long lNumObjects,
        [in] DWORD dwClientRefrVersion,
        [in, size_is(lNumObjects)]
            _WBEM_RECONNECT_INFO* apReconnectInfo,
        [in, out, size_is(lNumObjects)]
            _WBEM_RECONNECT_RESULTS* apReconnectResults,
        [out] DWORD* pdwSvrRefrVersion
    );
};


[
    restricted,
    object,
    uuid(423EC01E-2E35-11d2-B604-00104B703EFD)
]
interface IWbemWCOSmartEnum : IUnknown
{
    HRESULT Next(
        [in] REFGUID proxyGUID,
        [in] long lTimeout,
        [in] ULONG uCount,
        [out] ULONG* puReturned,
        [out] ULONG* pdwBuffSize,
        [out, size_is(,*pdwBuffSize)] byte** pBuffer
        );
};

[
    restricted,
    object,
    uuid(1C1C45EE-4395-11d2-B60B-00104B703EFD)
]
interface IWbemFetchSmartEnum : IUnknown
{
    HRESULT GetSmartEnum(
        [out] IWbemWCOSmartEnum** ppSmartEnum
    );
};

[
    restricted,
```

```
    object,
    uuid(d4781cd6-e5d3-44df-ad94-930efe48a887)
]
interface IWbemLoginClientID : IUnknown
{
    HRESULT SetClientInfo(
        [in, unique, string ] LPWSTR wszClientMachine,
        [in] long lClientProcId,
        [in] long lReserved
    );
};


[
    object,
    restricted,
    uuid(F309AD18-D86A-11d0-A075-00C04FB68820),
    pointer_default(unique)
]
interface IWbemLevel1Login :  IUnknown
{
    HRESULT EstablishPosition(
        [in, unique, string] wchar_t* wszLocaleList,
        [in] DWORD dwNumLocales,
        [out] DWORD* reserved
    );

    HRESULT RequestChallenge(
           [in, unique, string] wchar_t* wszNetworkResource,
        [in, unique, string] wchar_t* wszUser,
        [out, size_is(16), length_is(16)] unsigned char* Nonce
    );

    HRESULT WBEMLogin(
        [in, unique, string] wchar_t* wszPreferredLocale,
        [in, size_is(16), length_is(16), unique]
                unsigned char* AccessToken,
        [in] long lFlags,
        [in] IWbemContext* pCtx,
        [out] IWbemServices** ppNamespace
    );

    HRESULT NTLMLogin(
        [in, unique, string] LPWSTR wszNetworkResource,
        [in, unique, string] LPWSTR wszPreferredLocale,
        [in]   long lFlags,
        [in]   IWbemContext* pCtx,
        [out]  IWbemServices** ppNamespace
    );

};

[
    restricted,
    object,
    uuid(541679AB-2E5F-11d3-B34E-00104BCC4B4A)
]
interface IWbemLoginHelper : IUnknown
```

```
{
    HRESULT SetEvent(
        [in] const char * sEventToSet
    );
};
```

7 Appendix B: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows Server 2003 SP2
- Windows Server 2003
- Windows Vista
- Windows XP
- Windows XP SP1
- Windows XP 64-Bit Edition
- Windows 2000

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

<1> Section 2.2.1.6: The prefix "__" is specific to Windows and is not a CIM standard.

<2> Section 2.2.1.14: The Windows Client builds the IWBemContext object by using a set of specific IWBemContext methods to add, delete, and enumerate properties with their respective values. Note that the IWbemContext methods are not used by the protocol at any time. They are used internally by the client and the server to manage the content of the IWBemContext object.

<3> Section 2.2.1.15: This optimization technique is being used by Windows starting with Windows XP and Windows Server 2003.

<4> Section 2.2.1.15: The value MUST be 0 for Windows communication.

<5> Section 2.2.1.21: Windows uses the **m_dwProcessId** as the process identifier.

<6> Section 3.1.1.3: Windows tries to make the call at the highest authentication level RPC_C_AUTHN_PKT_PRIVACY and it gradually downgrades (decreasing the authentication level with one at every call) to RPC_C_AUTHN_NONE if the Distributed Component Object Model (DCOM) Remote Protocol (as specified in [MS-DCOM]) is unable to use the current authentication level.

<7> Section 3.1.5.1.1: Windows does not implement this method.

<8> Section 3.1.5.1.2: Windows does not implement this method and returns a WBEM_E_NOT_SUPPORTED error code.

<9> Section 3.1.5.1.3: Windows does not implement this method and returns an E_NOTIMPL error code.

<10> Section 3.1.5.2.18: Under Windows, this value MUST be "WQL", as specified in section 2.2.1.1.

<11> Section 3.1.5.2.19: Under Windows, this value MUST be "WQL," as specified in section 2.2.1.1.

<12> Section 3.1.5.2.20: Under Windows, this value MUST be "WQL", as specified in section 2.2.1.1.

<13> Section 3.1.5.2.21: Under Windows, this value MUST be "WQL", as specified in section 2.2.1.1.

<14> Section 3.1.5.5.1: To minimize network bandwidth, starting with Windows XP and Windows Server 2003, WMI uses an ObjectArray structure to send an array of CIM objects. The behavior is specified in section 2.2.1.15.

<15> Section 3.1.5.5.1: To minimize network bandwidth, starting with Windows XP and Windows Server 2003, the Windows Management Instrumentation Remote Protocol uses the ObjectArray structure to send an array of CIM objects. In this case, the server MAY replace the DCOM Remote Protocol marshaling and MUST send the ObjectArray structure by using the **IWbemObjectSink::Indicate** opnum.

<16> Section 3.1.5.5.2: Windows does not send progress information.

<17> Section 3.1.5.6: Windows implements and uses this interface starting with Windows XP and Windows Server 2003.

<18> Section 3.1.5.6.1: Windows implements and uses this interface starting with Windows XP and Windows Server 2003.

<19> Section 3.1.5.8: Windows XP, Windows Server 2003, and beyond implement this interface and ignore the lack of the interface on the server.

<20> Section 3.1.5.8.1: Windows client is calling that server interface; however, none of the parameters or the call made is taken into account by the server besides troubleshooting. Therefore, this interface has no effect on the protocol.

<21> Section 3.1.5.9: Windows NT, Windows 2000, Windows 2000 Server, Windows XP, and Windows XP SP1 implement this interface and ignore the lack of the interface on the server.

<22> Section 3.1.5.9.1: Windows client is calling that server interface; however, the call has no effect on the protocol.

<23> Section 3.1.5.10.1: Windows server requires an absolute file path starting with a drive letter.

<24> Section 3.1.5.10.2: Windows Server requires an absolute file path starting with a drive letter.

<25> Section 3.1.5.11: Windows implements the **IWbemBackupRestoreEx** interface in Windows XP and Windows Server 2003.

<26> Section 3.1.5.12: The **IWbemRefreshingServices** interface is only available on Windows XP, Windows Server 2003, and later.

<27> Section 3.1.5.12.4: Windows does not implement this method and returns WBEM_E_NOT_AVAILABLE error code.

<28> Section 3.1.5.13: The **IWbemRemoteRefresher** interface is only available on Windows XP, Windows Server 2003, and later.

<29> Section 3.1.5.13.3: The **Opnum5NotUsedOnWire** method is not used by Windows and therefore is not required for an implementation.

<30> Section 4.5: The optimized asynchronous delivery is supported by Windows XP and Windows Server 2003 and later.

<31> Section 5.1: Windows secures the access to each namespace and accepts only authenticated calls made at least at the RPC_C_AUTHN_LEVEL_CONNECT level. Windows behavior across different operating systems is specified in the following table.

Operating system version	Minimum authentication level
Windows NT	RPC_C_AUTHN_LEVEL_CONNECT
Windows 2000	RPC_C_AUTHN_LEVEL_CONNECT
Windows 2000 Professional SP3 and later	RPC_C_AUTHN_LEVEL_PKT
Windows XP	RPC_C_AUTHN_LEVEL_PKT
Windows Server 2003	RPC_C_AUTHN_LEVEL_PKT
Windows Vista	RPC_C_AUTHN_LEVEL_PKT

<32> Section 5.2: In Windows, local administrators are implicitly granted all rights in the table specified in section 5.2.

8 Index

L

M

N

O

P

Q

R

S

T

V

W