Document Identifier: DSP0223

Date: 2015-03-06

Version: 2.0.0

Generic Operations

Supersedes: 1.1

Document Type: Specification

Document Class: Normative

Document Status: Published

Document Language: en-US

Contents

Figures

Tables

Foreword

The *Generic Operations* specification (DSP0223) was originally prepared by the Generic Operations Working Group of the DMTF and is now owned by the Architecture Working Group of the DMTF.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. For information about the DMTF, see http://www.dmtf.org.

Acknowledgements

DMTF acknowledges the following individuals for their contributions to this specification:

- Jim Davis, WBEM Solutions
- George Ericson, EMC
- Steve Hand, Symantec
- Jon Hass, Dell
- Lawrence Lamers, VMware
- Andreas Maier, IBM (editor)
- Karl Schopmeyer, Inova Development

Document conventions

Typographical conventions

The following typographical conventions are used in this document:

- The titles of referenced documents are marked in *italics*.
- Important terms that are used for the first time are marked in *italics*.
- Generic parameters and generic types are marked in *italics*.
- The usage of terms typically links to their definition. Example: class path
- XML text is in `monospaced font`.

Experimental material

Experimental material has yet to receive sufficient review to satisfy the adoption requirements set forth by the DMTF. Experimental material is included in this document as an aid to implementers who are interested in likely future developments. Experimental material may change as implementation experience is gained. It is likely that experimental material will be included in an upcoming revision of the document. Until that time, experimental material is purely informational.

The following typographical convention indicates experimental material:

EXPERIMENTAL

Experimental material appears here.

EXPERIMENTAL

In places where this typographical convention cannot be used (for example, tables or figures), the "EXPERIMENTAL" label is used alone.

Generic Operations

1 Scope

DMTF defines a number of protocols that describe how managed resources that are modeled using CIM can be discovered, accessed and manipulated:

- CIM-XML: The protocol defined in the *CIM Operations over HTTP Specification* (DSP0200), the *Representation of CIM in XML Specification* (DSP0201) and the *DTD for Representation of CIM in XML* (DSP0203).
- WS-Management: The usage of the WS-Management protocol for CIM, as defined in the *WS-Management CIM Binding Specification* (DSP0227), the *WS-CIM Mapping Specification* (DSP0230), the *Web Services for Management Specification* (DSP0226), and other underlying Web Services specifications.
- CIM-RS: The RESTful protocol for CIM, as defined in *CIM-RS Protocol* (DSP0210) and in *CIM-RS Payload Representation in JSON* (DSP0211).
- SM-CLP: The protocol defined in the *Server Management Command Line Protocol Specification* (DSP0214), covering the core of the protocol common for all management profiles, and SM-CLP mapping specifications for each management profile, covering profile-specific aspects of the protocol such as verbs for extrinsic methods.

As different as these protocols are, they have certain operations and semantics in common, at least when looking at it from a higher level. These common semantics can be used to define generic operations. This specification defines an operational model and behavior associated to these operations at an abstracted, generic level, and common across these protocols.

The generic operations are expected to be used in the following areas:

- Future releases of management profiles can define requirements on intrinsic operations by referencing generic operations. Currently, they do that by referencing the operations defined for the CIM-XML protocol. Using generic operations allows management profiles to become independent of protocols. Management profiles defined in XML using the *Management Profile XML Schema* (DSP8028) are required to use generic operations.
- Future and existing DMTF protocols can define mappings between their protocol-specific operations and the generic operations. This drives more commonality across these protocols, and consequently makes it easier to support multiple protocols in client applications, server side instrumentation, and mapping bridges between protocols (also known as protocol gateways).
- Client APIs, server APIs and provider APIs can define their API calls conformant to the generic operations. This drives more commonality across these APIs and between these APIs and WBEM protocols, and consequently makes it easier to support multiple protocols with the same API in client libraries and server side instrumentation (e.g., provider APIs).

2 Normative references

The following referenced documents are indispensable for the application of this specification. For dated or versioned references, only the edition cited (including any corrigenda or DMTF update versions) applies. For references without a date or version, the latest published edition of the referenced document (including any corrigenda or DMTF update versions) applies.

DMTF DSP0004, *CIM Infrastructure Specification 2.8*,
http://www.dmtf.org/standards/published_documents/DSP0004_2.8.pdf

DMTF DSP0198, *WBEM Glossary 1.0*,
http://www.dmtf.org/standards/published_documents/DSP0198_1.0.pdf

DMTF DSP0207, *WBEM URI Mapping 1.0*,
http://www.dmtf.org/standards/published_documents/DSP0207_1.0.pdf

DMTF DSP0212, *Filter Query Language 1.0*,
http://www.dmtf.org/standards/published_documents/DSP0212_1.0.pdf

DMTF DSP1054, *Indications Profile 1.2*,
http://www.dmtf.org/standards/published_documents/DSP1054_1.2.pdf

DMTF DSP8016, *WBEM Operations Message Registry 1.1*,
http://schemas.dmtf.org/wbem/messageregistry/1/dsp8016_1.1.xml

ISO/IEC Directives, Part 2:2004, *Rules for the structure and drafting of International Standards*,
http://isotc.iso.org/livelink/livelink?func=ll&objId=4230456&objAction=browse

3 Terms and definitions

In this specification, some terms have a specific meaning beyond the normal English meaning. Those terms are defined in this clause.

The terms "shall" ("required"), "shall not", "should" ("recommended"), "should not" ("not recommended"), "may", "need not" ("not required"), "can" and "cannot" in this specification are to be interpreted as described in ISO/IEC Directives, Part 2, Annex H. The terms in parenthesis are alternatives for the preceding term, for use in exceptional cases when the preceding term cannot be used for linguistic reasons. ISO/IEC Directives, Part 2, Annex H specifies additional alternatives. Occurrences of such additional alternatives shall be interpreted in their normal English meaning.

The terms "clause", "subclause", "paragraph", "annex" in this specification are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 5.

The terms "normative" and "informative" in this specification are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 3. In this specification, clauses, subclauses or annexes indicated with "(informative)" as well as notes and examples do not contain normative content.

The terms "class path", "creation class", "instance path", "management profile", "namespace path", "object", "object path", "qualifier type path", "WBEM client", "client", "WBEM listener", "listener", "WBEM server", "server", "WBEM operation", "WBEM protocol", and any other terms defined in DSP0198 apply to this specification. The following additional terms are used in this document.

3.1
duplicate object

objects in a result set that have duplicate object paths.

3.2
duplicate object path

object paths in a result set that reference the same object accessible through the WBEM server.

3.3
effective qualifier value

The effective value of a qualifier specified on a schema element is the value that determines the qualifier behavior for the schema element, taking the qualifier propagation rules into account. For a complete definition, see DSP0004.

3.4

exposed elements of a class

The set of schema elements exposed by a class (i.e., properties and methods) is the union of the set of elements defined in the class (including overridden elements) and the set of inherited elements that are not overridden in the class. For a complete definition, see DSP0004.

3.5

generic listener operation

a generic operation directed from a WBEM server to a WBEM listener. Also called listener operation. For details, see 5.1.

3.6

generic operation

a generic operation as defined in this specification. Also called operation. They are divided into generic listener operations and generic server operations. For details, see 5.1.

3.7

generic operation request

the request portion of a generic operation. Also called operation request. For details, see 5.1.

3.8

generic operation response

the response portion of a generic operation. Also called operation response. For details, see 5.1.

3.9

generic operations mapping

a mapping of generic operations to the operations of some other protocol (e.g., WBEM operations) or to the calls of some API, as defined in 5.2.

3.10

generic server operation

a generic operation directed from a WBEM client to a WBEM server. Also called server operation. For details, see 5.1.

3.11

isolation

the set of behaviors that describe how the execution of an operation affects the execution of another, concurrent operation, as defined in 5.8.4.

3.12

volatile property

a property in an instance whose value may change as a WBEM client obtains the instance repeatedly without performing any client-originated updates to the property value.

3.13

WBEM listener operation

a WBEM operation that is originated on a WBEM server and processed by a WBEM listener. For details, see 5.1.

3.14

WBEM protocol mapping

a mapping of generic operations to a WBEM protocol, as defined in 5.2.

3.15
WBEM server operation
a WBEM operation that is originated by a WBEM client and processed by a WBEM server. For details, see 5.1.

4 Symbols and abbreviated terms

The abbreviations "API", "CIM", "CIM-XML", "CIM-RS", "CQL", "UML", "WBEM", "WS-Management", "XML", and any other symbols and abbreviations defined in DSP0198 apply to this specification. The following additional abbreviations are used in this document.

4.1
SM-CLP

Server Management Command Line Protocol, defined in DSP0214

5 Concepts

This clause defines concepts that are the basis for the definition of the generic operations.

5.1 Interaction model for generic operations

Generic operations are divided into two categories:

- **Generic server operations:** An operation request is sent from a WBEM client to a WBEM server in order to initiate the processing of the operation, and an operation response is sent back from the server to the client upon completion of the operation.
- **Generic listener operations:** An operation request is sent from a WBEM server to a WBEM listener in order to initiate the processing of the operation, and an operation response is sent back from the listener to the server upon completion of the operation.

Figure 1 shows the interaction model for generic server operations, using a UML sequence diagram:

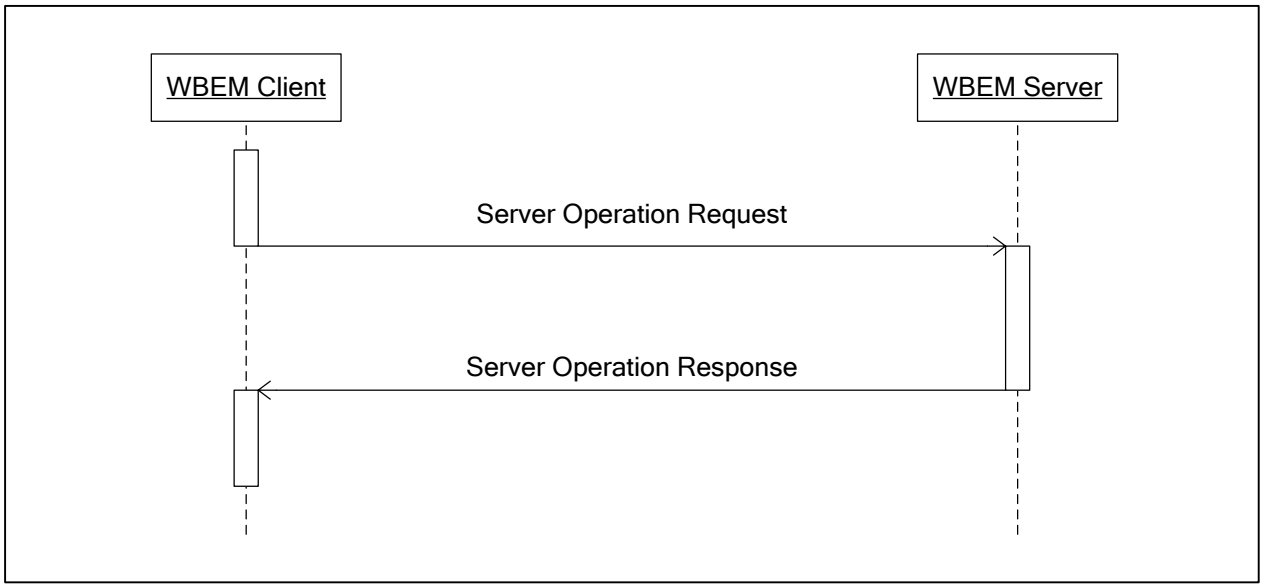

Figure 1 – Interaction model for generic server operations

Figure 2 shows the interaction model for generic listener operations, using a UML sequence diagram:

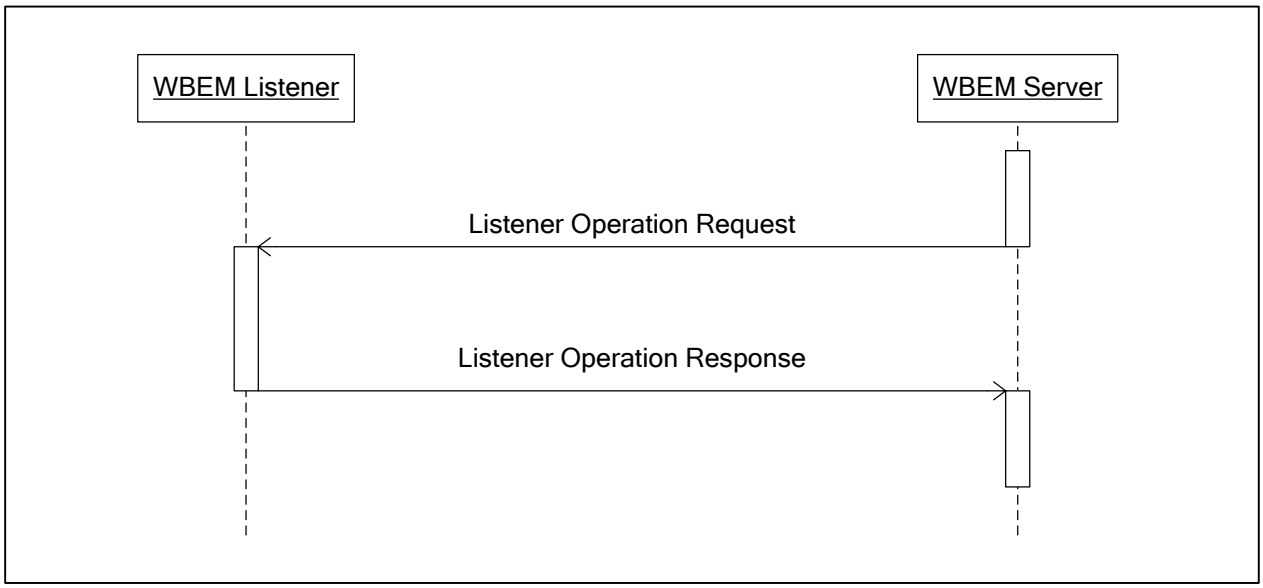

Figure 2 – Interaction model for generic listener operations

The operation request and operation response at the level of generic operations do not necessarily need to correspond directly to WBEM operations, that is to messages that are flowing at the level of the WBEM protocol. For example, a generic operation response may be delivered asynchronously at the level of the WBEM protocol.

At the level of generic operations, any *input parameters* are part of the operation request, and any *output parameters* are part of the operation response. A WBEM protocol may choose to do that differently, for example by pushing some of the input parameters to the server in the form of options that are set, and that are used during the processing of subsequent operations.

This abstraction of generic operations from WBEM operations allows keeping the definition of the generic operations simple and scoped to defining the operation semantics. The details about the actual message flows are left to the scope of WBEM protocols. This separation is key in order to use the same definition of generic operations for multiple WBEM protocols.

5.2 Generic operations mappings

5.2.1 Overview

Figure 3 shows mappings of generic operations to WBEM protocols and APIs. These mappings allow determining which WBEM operations or API calls need to be implemented for a particular generic operation to be supported. This is used for example when implementing management profiles that specify provisions for intrinsic operations by referencing generic operations.

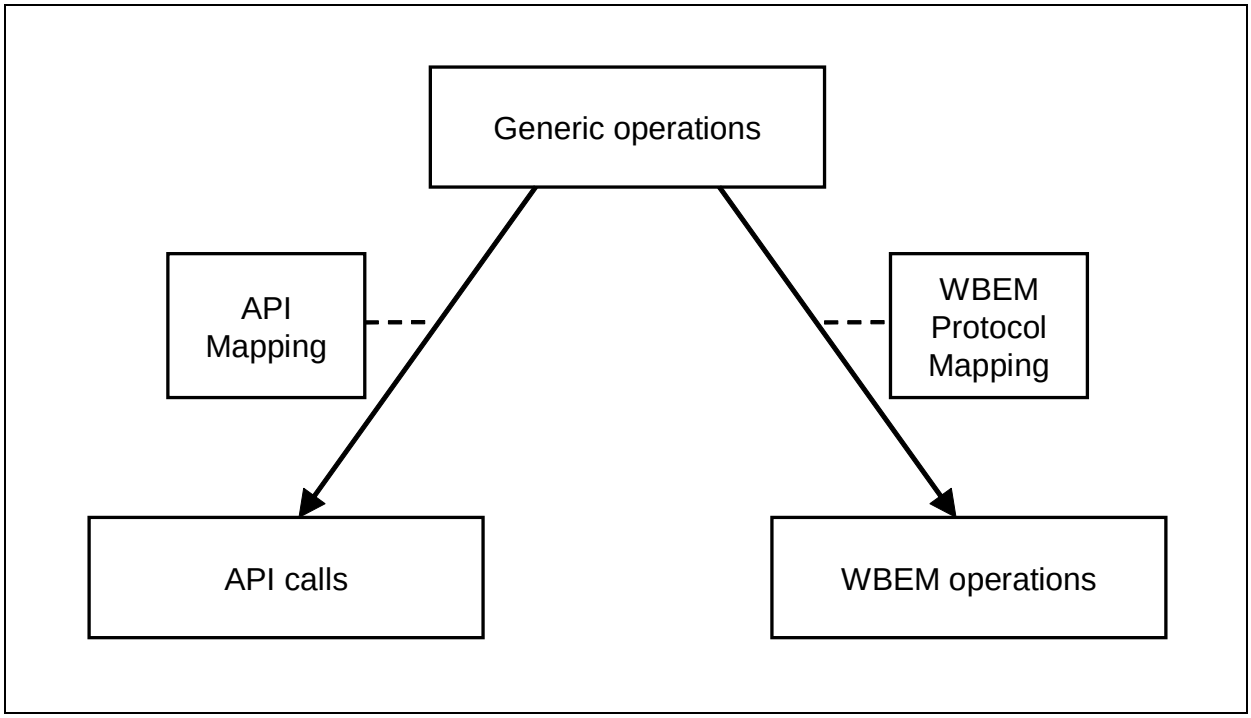

Figure 3 – Generic operations mappings

5.2.2 Recommendations

This subclause provides recommendations for specifying WBEM protocol mappings and API mappings that provide for determining the WBEM operations or API calls that support a particular generic operation, and specify conformance.

There is no requirement that WBEM protocol mappings and API mappings are defined in a separate specification (i.e., they can be defined in the specifications that define the WBEM protocol or API).

The following recommendations apply:

- WBEM protocol mappings and API mappings should define the mapping from a perspective of the generic operation (i.e., by listing the relevant generic operation at the top level).
- For each generic operation listed in the mapping, the corresponding WBEM operations or API calls should be stated that provide the functionality supporting the generic operation.
- For each parameter defined for a generic operation listed in the mapping, the corresponding parameters and return values of the WBEM operations or API calls should be stated.
- A statement should be made for each generic operation as to whether or not the operation is supported in a conformant way, as defined in 5.3.2. If the operation is supported in a non-conformant way, the deviations should be stated.
- A statement should be made for the entire WBEM protocol or API as to whether or not it is conformant to generic operations.

5.3 Conformance to generic operations

Conformance of a WBEM protocol or API to generic operations is defined at two levels:

1) At the level of the entire WBEM protocol or API
2) At the level of single WBEM operations or single API calls

The guiding principle for conformance to generic operations is that a WBEM protocol or API call is able to completely represent the generic operations and their semantics. Functionalities of the WBEM protocol or API that go beyond the functionality of generic operations are not relevant for conformance.

5.3.1 Conformance of entire WBEM protocols or APIs

A WBEM protocol or API is conformant to generic operations if all generic operations defined in this specification are supported by WBEM operations or API calls in a conformant way, as defined in 5.3.2.

Conformant WBEM protocols or APIs may define WBEM operations or API calls in addition to those that are mapped to generic operations.

5.3.2 Conformance of single WBEM operations or API calls

A particular generic operation is supported by WBEM operations or API calls in a conformant way if all of the following is satisfied:

- The generic operation has one or more corresponding WBEM operations or API calls that provide the functionality of the generic operation. The names of these corresponding WBEM operations or API calls may be different from the name of the generic operation.
- Functionalities that are required to be supported for a generic operation are supported by the corresponding WBEM operations or API calls with the semantics defined by the generic operation.
- If functionalities that are optional to be supported for a generic operation are supported by the corresponding WBEM operations or API calls, they are supported with the semantics defined by the generic operation.
- Each parameter of a generic operation is mapped to one or more corresponding parameters of the corresponding WBEM operations or API calls
- For each parameter of a generic operation, the provisions defined in 5.3.3 are satisfied.

WBEM operations or API calls that support a generic operation in a conformant way, may support parameters or return values in addition to the parameters mapped to parameters of the corresponding

generic operation. Defining additional parameters can affect the ability to transform one WBEM protocol into another (e.g. in protocol gateways).

5.3.3 Requirement levels for operation parameters

The parameters defined for generic operations each have a requirement level, as defined in this subclause.

A conformant WBEM protocol or API shall delegate these requirement levels to the receiver of the WBEM operation (for example, to the WBEM server for a server operation, and to the WBEM listener for a listener operation), as follows:

Mandatory

For operation parameters designated as mandatory, conformant WBEM protocols and APIs shall document the support of the corresponding operation parameters by the receiver of the WBEM operation as:

- Mandatory.

Conditional

For operation parameters designated as conditional, conformant WBEM protocols and APIs shall document the support of the corresponding operation parameters by the receiver of the WBEM operation as one of:

- Mandatory, if the condition can be evaluated at the time the WBEM protocol or API specification is written and the condition is met,
- Optional, if the condition can be evaluated at the time the WBEM protocol or API specification is written and the condition is not met,
- Conditional, potentially with a different condition, if the condition cannot be evaluated at the time the WBEM protocol or API specification is written.

Optional

For operation parameters designated as optional, conformant WBEM protocols and APIs shall document the support of the corresponding operation parameters by the receiver of the WBEM operation as one of:

- Mandatory,
- Optional,
- Conditional.

In all those cases, conformant WBEM protocols and APIs may specify that supplying values for a supported parameter is optional as long as the protocol or API defines a default value for the parameter. In other words, there are two different kinds of requirements related to parameters:

1) The requirement defined by this document to support a parameter in a WBEM protocol or API by the receiver of an operation
2) The requirement defined by the WBEM protocol or API specification for supplying a value for a supported parameter by the originator of an operation

5.4 Generic types

This specification defines the following generic data types for use by operation parameters of generic operations.

5.4.1 CIM data types

All CIM data types defined in DSP0004 (e.g., boolean) may be used as generic types. Values of these data types can assume the (untyped) value NULL, as defined in DSP0004.

5.4.2 NamespacePath

A value of the generic type *NamespacePath* represents a namespace path as defined in DSP0004.

This specification does not define particular sub-components of a namespace path; as a result, any requirements on the presence of such sub-components are left to conformant WBEM protocols.

Conformant WBEM protocols shall support all characteristics of *NamespacePath* values and may support additional characteristics.

5.4.3 InstancePath

A value of the generic type *InstancePath* represents an instance path as defined in DSP0004.

An *InstancePath* value shall specify the class name and key binding components of the represented instance path. Any requirements for specifying or omitting the namespace path component in an *InstancePath* value are left to conformant WBEM protocols.

Conformant WBEM protocols shall support all characteristics of *InstancePath* values and may support additional characteristics.

5.4.4 ClassPath

A value of the generic type *ClassPath* represents a class path as defined in DSP0004.

A *ClassPath* value shall specify the class name component of the represented class path. Any requirements for specifying or omitting the namespace path component in a *ClassPath* value are left to conformant WBEM protocols.

Conformant WBEM protocols shall support all characteristics of *ClassPath* values and may support additional characteristics.

5.4.5 QualifierTypePath

A value of the generic type *QualifierTypePath* represents a qualifier type path as defined in DSP0004.

A *QualifierTypePath* value shall specify the qualifier name component of the represented qualifier type path. Any requirements for specifying or omitting the namespace path component in a *QualifierTypePath* value are left to conformant WBEM protocols.

Conformant WBEM protocols shall support all characteristics of *ClassPath* values may support additional characteristics.

5.4.6 InstanceSpecification

A value of the generic type *InstanceSpecification* is a representation of a CIM instance as defined for the *Instance* meta-element defined in DSP0004, containing:

- name of the creation class of the instance
- all or a subset of the static and non-static properties exposed by the creation class of the instance

Each property in an *InstanceSpecification* shall contain:

- name of the property
- value of the property
- optional: Class origin of the property
- optional: Data type of the property

InstanceSpecification does not contain the instance path of the instance, because there are some situations in which the instance data is needed without an instance path. The *InstanceSpecificationWithPath* type is used when the instance path is needed in addition to the instance data.

Generic operations using this type define the rules for the optional items in the content of this type.

5.4.7 ClassSpecification

A value of the generic type *ClassSpecification* is a representation of a CIM class as defined for the *Class* meta-element defined in DSP0004, containing:

- name of the class
- name of the superclass, if any
- all or a subset of the static and non-static properties (that is, the property definitions) exposed by the class. As defined in DSP0004, the set of properties exposed by a class includes any properties inherited from superclasses, where overridden properties are included only once.
- all of the static and non-static methods exposed by the class. As defined in DSP0004, the set of methods exposed by a class includes any methods inherited from superclasses, where overridden methods are included only once.
- optional: all of the qualifiers exposed by the class that are defined on the class or any of its superclasses

Each property in a *ClassSpecification* shall contain:

- name of the property
- data type of the property
- default value of the property
- optional: all of the qualifiers exposed by the property that are defined on the property or any of its overridden properties

Each method in a *ClassSpecification* shall contain:

- name of the method
- data type of the return value of the method
- all of the parameters of the method
- optional: all of the qualifiers exposed by the method that are defined on the method or any of its overridden methods

Each parameter in that method shall contain:

- name of the parameter
- data type of the parameter
- optional: all of the qualifiers exposed by the parameter that are defined on the parameter or the corresponding parameter in any of its overridden methods

Each qualifier in any of the items above shall contain:

- name of the qualifier
- effective value of the qualifier, as seen in the scope of the class represented by *Class*

ClassSpecification does not contain the class path of the class. The *ClassSpecificationWithPath* type is used when the class path is needed in addition to the class.

Generic operations using this type define the rules for the optional items in the content of this type.

5.4.8 QualifierType

A value of the generic type *QualifierType* is a representation of a CIM qualifier type as defined for the *QualifierType* meta-element defined in DSP0004 (i.e., a qualifier declaration) containing:

- name of the qualifier
- data type of the qualifier
- default value of the qualifier
- all flavors of the qualifier
- all scopes of the qualifier

QualifierType does not contain the qualifier type path of the qualifier type. The *QualifierTypeWithPath* type is used when the qualifier type path is needed in addition to the qualifier type.

5.4.9 InstanceSpecificationWithPath

A value of the generic type *InstanceSpecificationWithPath* combines the content of *InstanceSpecification* and *InstancePath*.

InstanceSpecification shall represent the instance referenced by *InstancePath*.

5.4.10 ClassSpecificationWithPath

A value of the generic type *ClassSpecificationWithPath* combines the content of *ClassSpecification* and *ClassPath*.

ClassSpecification shall represent the class referenced by *ClassPath*.

5.4.11 QualifierTypeWithPath

A value of the generic type *QualifierTypeWithPath* combines the content of *QualifierType* and *QualifierTypePath*.

QualifierType shall represent the qualifier type referenced by *QualifierTypePath*.

5.4.12 ClassName

A value of the generic type *ClassName* is the name of a CIM class, including its schema prefix.

5.4.13 PropertyName

A value of the generic type *PropertyName* is the name of a CIM property or reference.

The class defining the property is not identified by the data in this type.

5.4.14 MethodName

A value of the generic type *MethodName* is the name of a CIM method.

The class defining the method and the method signature are not identified by the data in this type.

5.4.15 ParameterValue

A value of the generic type *ParameterValue* is a parameter value used as an input or output parameter during invocation of a CIM method, containing:

- name of the parameter
- value of the parameter
- optional: Data type of the parameter

Generic operations using this type define the rules for the optional items in the content of this type.

5.4.16 ReturnValue

A value of the generic type *ReturnValue* is the value returned by the invocation of a CIM method, containing:

- return value
- optional: Data type of the return value

Generic operations using this type define the rules for the optional items in the content of this type.

5.4.17 QueryString

A value of the generic type *QueryString* is a query string in some query language. The query language is not identified by the data in this type.

5.4.18 QueryLanguage

A value of the generic type *QueryLanguage* is a query language of a query string.

5.4.19 EnumerationContext

A value of the generic type *EnumerationContext* is a value that uniquely identifies an enumeration session used in pulled instance enumeration operations. It is opaque to WBEM clients.

5.4.20 ListenerDestination

A value of the generic type *ListenerDestination* is a value that uniquely addresses a WBEM listener for purposes of delivering an indication to it using the *DeliverIndication* operation (see 6.9.1).

The format of the address is defined by the WBEM protocol.

5.5 Success and failure

All generic operations either succeed or fail. There is no concept of "partial success".

If a generic operation succeeds, it delivers its output data back to the operation requester, and does not include any error messages.

If it fails, it delivers back one or more error messages, and no output data. For details about error messages, see 5.7.

For example, if an instance enumeration operation were able to return some instances successfully, but not all successfully, then the operation shall fail without returning any instances.

The WBEM operations mapped to generic operations by a conformant WBEM protocol shall also either succeed or fail, as described above.

5.6 Preconditions and postconditions

Each generic operation specifies a set of zero or more preconditions and a set of zero or more postconditions.

Each precondition in the set needs to be satisfied for the operation to be able to succeed. If one or more preconditions are not satisfied, the operation shall fail, indicating the unsatisfied precondition using a generic error message from the set listed for the operation that describes the unsatisfied precondition.

A successful execution of the generic operation shall guarantee that all postconditions in the set are satisfied.

5.7 Generic error messages

Each generic operation specifies a set of generic error messages. These generic error messages are DMTF standard messages (see DSP0228) from the WBEM Operations Message Registry (DSP8016). Each error message from this registry describes a particular error situation.

A conformant WBEM protocol shall support error handling in the following way:

- Return DMTF standard messages (also known as "extended error handling").
  In this case, the WBEM operation shall return the generic error message defined for the generic operation that matches the error situation, and may return additional error messages.

The other alternatives permitted in version 1 of this document are no longer permitted.

The generic error messages specified for each generic operation have a requirement level defined in context of that operation.

A conformant WBEM protocol or API shall delegate these requirement levels to the receiver of the WBEM operation (for example, to the WBEM server for a server operation, and to the WBEM listener for a listener operation), as follows.

The allowable requirement levels for generic error messages in the context of a generic operation are:

Mandatory

For generic error messages designated as mandatory, conformant WBEM protocols and APIs shall document the error message as mandatory to be supported by the receiver of the WBEM operation.

Conditional

For generic error messages designated as conditional, conformant WBEM protocols and APIs shall document the support of the error message by the receiver of the WBEM operation as one of:

- Mandatory, if the condition can be evaluated at the time the WBEM protocol or API specification is written and the condition is met,
- Optional, if the condition can be evaluated at the time the WBEM protocol or API specification is written and the condition is not met,
- Conditional, potentially with a different condition, if the condition cannot be evaluated at the time the WBEM protocol or API specification is written.

Optional

For generic error messages designated as optional, conformant WBEM protocols and APIs shall document the support of the error message by the receiver of the WBEM operation as one of:

- Mandatory,
- Optional,
- Conditional.

Each generic operation designates one of its input parameters to be a "context parameter." The messages defined in the WBEM Operations Message Registry (DSP8016)) may include name and value of the context parameter in order to provide information about the invocation context.

This specification does not define any order or precedence for generic error messages to be returned by generic operations. This implies that the order in which the generic error messages are listed in the description of each generic operation has no binding significance on the order in which a conformant WBEM protocol would need to apply any tests to surface these errors, nor does the documented order require a precedence of error messages. However, the order in which the generic error messages are listed is meant to give some guidance about a typical order of precedence.

WBEM clients shall be prepared to deal with all generic error messages that are listed for a generic operation.

5.8 Consistency model

This subclause defines consistency requirements for generic operations.

Conformant WBEM protocols shall conform to the rules defined in this subclause for the WBEM operations to which the supported generic operations are mapped. WBEM protocols may define additional constraints for WBEM operations.

This specification does not define responsibilities for detecting violations to these rules.

5.8.1 Definition of ACID properties

This subclause defines atomicity, consistency, isolation and durability (ACID) properties for use by generic operations defined in this specification and by management profiles (see DSP1001).

Each generic operation defines requirements on its ACID properties. Management profiles that use generic operations to state their operation requirements inherit these requirements on ACID properties and may specify additional requirements. Profiles should not remove or weaken requirements on ACID properties defined by generic operations.

5.8.1.1 Atomicity

Operations and methods are considered *atomic* if and only if their effects on the managed resources and on CIM instances either occur completely or not at all.

Atomicity only applies to operations and methods that modify the managed resources or CIM instances through the management interface.

5.8.1.2 Update consistency

Operations and methods are considered *update-consistent* if and only if the managed resources and CIM instances are never left in an inconsistent state after a modification.

What constitutes a consistent state is defined in DSP0004 and in management profiles.

Update consistency only applies to operations and methods that modify the managed resources or CIM instances through the management interface.

5.8.1.3 Isolation

Operations and methods are considered *isolated* if and only if their results and their effects on the managed resources and on CIM instances appear to be serialized with the results and effects of any other operations and methods, as observed through the management interface.

Isolation applies to operations and methods that retrieve information through the management interface, and to operations that modify the managed resources or CIM instances through the management interface.

5.8.1.4 Durability

Operations and methods are considered *durable* if and only if their effects on the managed resources and on CIM instances will not be undone, other than by some other action that may or may not be caused through the profile defined management interface.

Durability only applies to operations and methods that modify the managed resources or CIM instances through the management interface.

5.8.2 Time consistency within instance representations

The property values of an instance representation returned by any generic operation shall represent a snapshot of the instance object that exists in the server.

If a WBEM protocol provides the capability to transfer an operation response in multiple parts, and a particular instance representation is distributed over multiple parts of the response which are transferred at different points in times, the property values of that instance representation still need to satisfy the time consistency constraint.

5.8.3 Staleness of information returned

Conformant WBEM protocols should define that implementations should do a best effort to return the most current information, as far as property values of instances and also the existence of instances are concerned.

5.8.4 Isolation between operations

This specification defines no particular requirements regarding isolation between operations in addition to the other consistency rules defined in 5.8.

For example, if an instance is deleted and after that another one is created, an enumeration operation executed concurrently may consistently include the instance that got deleted just before that happened, as well as the new instance after it got consistently created, hence returning a set of instances that never existed at the same time. This example satisfies all consistency rules defined in this specification. An example where other consistency rules determine the overall behavior is a GetInstance operation executing concurrently with a

ModifyInstance operation on the same instance. The consistency rules defined in 5.8.2 require that this GetInstance operation needs to return an instance representation that either has none or all of the modifications requested by the ModifyInstance operation.

5.8.5 Duplicate return of CIM objects or object paths

Any generic operations returning CIM object representations or CIM object paths should not return duplicate objects or duplicate object paths.

If duplicate objects or duplicate object paths are returned, WBEM clients should consider the last occurrence of a duplicate object or duplicate object path in the sequence as the valid occurrence to work with, and should ignore all other duplicate occurrences.

DSP0004 requires that a CIM namespace in a WBEM server does not contain duplicate objects (i.e., instances, classes, qualifier types) at any point in time. However, given the rule above, the result set of a generic operation may.

An example for a situation in which duplicate instances or instance paths might be returned is a sequence of instance deletion and creation with the same key values concurrently to an enumeration operation, all in the same namespace.

As a consequence, a WBEM server is not obliged to test for, correct or reject any duplicate objects or object paths in the result set of an operation.

5.8.6 Time consistency between returned CIM objects

This specification does not mandate any time consistency between the CIM objects or CIM object paths returned by generic operations.

For example, if a WBEM server processes an instance enumeration operation by contacting multiple independent infrastructure components each of which contributes instances to the combined result set, the result set may contain instance representations that represent different points in time.

However, the rule defined in 5.8.2 requires that consistency is maintained within each single instance representation.

5.8.7 Order of returned CIM objects

For operations that do not support the specification of a sort order, the order of returned CIM objects is implementation-dependent.

For example, if a WBEM server processes an instance enumeration operation by contacting multiple independent infrastructure components each of which contributes instances to the combined result set, the resulting order might be an arbitrary merge of the sequences of instances contributed by each component.

WBEM protocols may define additional requirements on the order of returned CIM objects.

5.8.8 Validity of returned object paths

This specification does not mandate that object paths returned to a WBEM client are still valid by the time the WBEM client attempts to use them in subsequent operations in order to address those objects.

For example: if a WBEM server returns an instance path and an operation then deletes the instance, a subsequent attempt to get the instance using the returned instance path will fail.

5.8.9 Effects of deleting an instance

Deleting an instance may affect the overall consistency because other instances depend on the instance to be deleted. Instances that depend on the instance to be deleted are called "dependent instances" in this specification.

The behavior of operations that delete instances (such as *DeleteInstance*) cannot be defined in a generally applicable way. The following options are available for defining the handling of the deletion of an instance in the presence of dependent instances (e.g., in management profiles or in the CIM schema):

- **Delete propagation:** Delete any dependent instances implicitly along with the instance to be deleted.

  Specifications using this specification need to give particular consideration to circular dependencies when defining rules for propagating deletion.

  NOTE Such dependent instances may reside in a different namespace (which may reside in a different WBEM server) than the instance to be deleted.

- **Rejection:** Reject the deletion of the instance to be deleted, leaving it to the WBEM client to delete dependent instances first.

The following options are **not** available for defining the handling of the deletion of an instance in the presence of dependent instances:

- **Deletion without propagation:** Delete the instance to be deleted but do not delete any dependent instances. This would cause an inconsistent state in the model.

The following instances are considered dependent instances for this purpose:

- **Composition:** Instances associated to an instance to be deleted, via a composition where the instance to be deleted is on the aggregate side.

  The definition of the *Composition* qualifier in DSP0004 requires that this case is handled by propagating the deletion of the aggregate instance to any aggregated instances and their composition instances.

- **Key propagation:** Instances of classes that have propagated keys (key properties exposing a value of TRUE for the *Propagated* qualifier, i.e., weak instances) are considered dependents of the instance from which the keys propagate (i.e., the strong instance).

  The definition of the *Propagated* qualifier in DSP0004 requires that this case is handled by propagating the deletion of the strong instance to any weak instances and their association instances.

- **Referencing associations:** Association instances that reference the instance to be deleted.

  This case shall be handled with any or a combination of the following options:

  – by propagating the deletion of the referenced instance to its referencing association instance

  – by rejecting the deletion of the referenced instance to be deleted.

- **Qualifier defined delete propagation:** Instances to be deleted as a result of *IfDelete* and *Delete* qualifiers, as defined in DSP0004.

  Support of the *IfDelete* and *Delete* qualifiers by a WBEM server is optional, as defined in DSP0004.

  This concept can be used to propagate deletion from an instance to its referencing association instance, from an association instance to its referenced instances, and in combination also between associated instances.

  The definition of the *IfDelete* and *Delete* qualifiers in DSP0004 requires that this case is handled by propagating the deletion of an instance to which the *IfDelete* qualifier applies, to any instances to which the corresponding *Delete* qualifier applies.

- **Multiplicity underflow:** Instances associated to an instance to be deleted via an association with a minimum multiplicity (as defined with *Min* qualifier in the schema, or as constrained by management profiles) larger than 0 on the reference to the instance to be deleted, if the deletion would violate the minimum multiplicity that is required.

  EXAMPLE: Association AB references class A with *Min (2)* and references class B. Therefore, each instance of B is supposed to be associated via AB with least two instances of A. If an instance of A is to be deleted, and there is only one other instance of A associated to the instance of B that is associated with the instance of A to be deleted, the minimum multiplicity would be violated by the deletion.

  This case shall be handled with any or a combination of the following options:

  – by propagating the deletion of the instance to be deleted to its associated instance defining the multiplicity constraint, and the association instance.

  – by rejecting the original deletion.

6 Generic operations

This clause defines the generic operations. They are listed in Table 1, grouped by their headings.

Table 1 – List of generic operations

Group	Generic Operation	Description
Instance operations	GetInstance	See 6.3.1
	DeleteInstance	See 6.3.2
	ModifyInstance	See 6.3.3
	CreateInstance	See 6.3.4
Instance enumeration operations	OpenEnumerateInstances	See 6.4.3
	OpenAssociators	See 6.4.4
	OpenReferences	See 6.4.5
	OpenQueryInstances	See 6.4.6
	PullInstancesWithPath	See 6.4.8
	PullInstances	See 6.4.9
	CloseEnumeration	See 6.4.10
Method invocation operations	InvokeMethod	See 6.5.1
	InvokeStaticMethod	See 6.5.2

Group	Generic Operation	Description
Class operations	GetClass	See 6.6.1
	DeleteClass	See 6.6.2
	ModifyClass	See 6.6.3
	CreateClass	See 6.6.4
Class enumeration operations	EnumerateClasses	See 6.7.1
	AssociatorClasses	See 6.7.2
	ReferenceClasses	See 6.7.3
Qualifier type operations	GetQualifierType	See 6.8.1
	DeleteQualifierType	See 6.8.2
	ModifyQualifierType	See 6.8.3
	CreateQualifierType	See 6.8.4
	EnumerateQualifierTypes	See 6.8.5

6.1 Description format

The generic operations are described using the following format. Items in angle brackets (e.g., "<name>") need to be replaced by some other text, as described further down in this subclause.

Purpose:

<Short description of the purpose of the operation.>

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
<diname>	<ditype>	<direq>	<Description of the operation parameter, including any conditions for requirement level Conditional> <The text "(Context Parameter)" for the parameter that is supposed to be displayed in messages, as defined in 5.7>
…	…	…	…

Operation Output Parameters:

Generic Name	Generic Type	Requirement	Description
<diname>	<ditype>	<direq>	<Description of the operation parameter, including any conditions for requirement level Conditional>
…	…	…	…

Description:

<A detailed description of the semantics of the operation including all conditions and behaviors except those listed under Preconditions and Postconditions>

Preconditions:

- <List of additional preconditions for the operation, in plain text. Preconditions pertain to the state before an operation gets invoked. They have nothing to do with the execution of the operation or any effects the operation causes. They represent the conditions that are required to be met in order for the operation to have a chance to execute successfully. Although not required for preconditions, this specification uses "*shall*" to specify preconditions.>

Postconditions:

- <List of additional postconditions for the operation, in plain text. Postconditions describe the state after an operation has been executed successfully. In other words, they represent the guarantees an implementation needs to give in the case of successful execution.>

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
<msgid>	<msgname>	<msgreq>	<msgsrc>	<Any description in addition to the description in the message registry>
...	**...**	**...**		**...**

The items in angle brackets that are not already described in the format above, have the following meaning:

<diname> Generic name of the operation parameter.

<ditype> Generic type of the operation parameter, as defined in 5.4.

<direq> Requirement level of the operation parameter, as defined in 5.3.3.

<msgid> Message ID of the message, as defined in a DMTF message registry. The message ID is the concatenation of the values of the XML attributes `MESSAGE/MESSAGE_ID@PREFIX` and `MESSAGE/MESSAGE_ID@SEQUENCE_NUMBER`.

<msgname> Message name of the message, as defined in a DMTF message registry. The message name is the value of the XML attribute `MESSAGE@NAME`.

<msgreq> Requirement level of the message, as defined in 5.7.

<msgsrc> Sources of the message. One or more values may be specified. Valid values are:

Infrastructure – the message is implemented by the common infrastructure portion of the WBEM server.

Class implem. – the message is implemented by the class specific portion of the WBEM server.

The message sources information is a recommendation only, for implementations of a WBEM server that distinguish between a common infrastructure portion (e.g., CIMOM) and class specific portion (e.g., providers).

6.2 Common operation parameters for all operations

This subclause defines commonly used operation parameters for the operations. The description of the individual operations references these operation parameters as appropriate. However, not every operation uses every one of these operation parameters.

6.2.1 IncludeQualifiers

The *IncludeQualifiers* operation input parameter controls whether qualifier values are returned for any returned CIM element in any returned class of a class operation.

Support for the *IncludeQualifiers* operation parameter in a conformant WBEM protocol is mandatory.

If *IncludeQualifiers* is TRUE, then any returned class and any returned CIM element within each returned class shall contain qualifier values for those qualifiers that have a value different from the default value defined in the declaration of the qualifier type. Any other qualifier values should not be included.

NOTE In order to inspect the scope and default value of any qualifiers that are not included in the returned class, a WBEM client can use operation *EnumerateQualifierTypes* to retrieve the qualifier type declarations that exist in a namespace.

If *IncludeQualifiers* is FALSE, then any returned class and any returned CIM element within each returned class shall not contain any qualifier values.

6.2.2 <element>List

The operation output parameters *InstanceList*, *InstancePathList*, *ClassList*, *ClassPathList*, and *QualifierTypeList* contain a sequence of elements, and are referred to as the *result set* of the operation.

The sequence is ordered in the sense that there is a relation of "before" and "after" between elements in the sequence and the sequence has a beginning and an end. However, this does not imply that the sequence is sorted according to some criteria.

Clause 5.8 defines rules for dealing with duplicate objects or duplicate object paths in the result set of an operation.

6.3 Instance operations

This subclause defines server operations that target a single instance, or create an instance.

6.3.1 GetInstance

Purpose:

Retrieves an instance.

874 ## Operation Input Parameters:

875

Generic Name	Generic Type	Requirement	Description
InstancePath	InstancePath	Mandatory	Instance path of the instance to be retrieved (Context Parameter)
IncludedProperties	PropertyName [ ]	Optional	NULL, or unordered set of property names, acting as a restricting filter on the properties included in the returned instance

876

877 ## Operation Output Parameters:

878

Generic Name	Generic Type	Requirement	Description
Instance	InstanceSpecification	Mandatory	Representation of the retrieved instance

879

880 ## Description:

881 The *GetInstance* operation retrieves a representation of the instance referenced by *InstancePath*.

882 As defined in the description of the *InstancePath* type, the instance path of the instance to be
883 retrieved is interpreted in a non-polymorphic way, i.e., it references the specified instance only and
884 does not include any instances with the same key values in subclasses.

885 The set of properties to be included in the retrieved instance shall be determined using the following
886 algorithm:

887 • Initially, the set of properties to be included is the set of properties exposed by the creation
888 class of the instance. This includes all the duplicates of any duplicate non-overridden
889 properties.

890 • If the *IncludedProperties* operation input parameter is supported by the WBEM protocol
891 and if its value is not NULL, it acts as a restricting filter on the properties to be included in
892 the returned instance representation such that any properties exposed by the creation
893 class of the instance that are not named in that operation parameter are removed from the
894 set of properties to be included. Any duplicate or invalid property names in the
895 *IncludedProperties* operation input parameter shall be ignored. A non-NULL empty
896 *IncludedProperties* list removes all properties from the set of properties to be included.

897 • Conformant WBEM protocols may specify rules that cause properties with a value of NULL
898 to be removed from the set of properties to be included.

899 ## Preconditions:

900 • The instance referenced by *InstancePath* shall exist. If it does not exist, the operation shall fail,
901 indicating WIPG0213.

902 • The creation class of the instance referenced by *InstancePath* shall exist. If it does not exist, the
903 operation shall fail, indicating WIPG0214.

904 • The namespace of the instance referenced by *InstancePath* shall exist. If it does not exist, the
905 operation shall fail, indicating WIPG0204.

906 **Postconditions:**

907 • The instance representation shall have been returned with the properties as defined in the
908 Description paragraph for this operation.

909 • Requirements on ACID properties:

910 – Atomicity: N/A

911 – Update Consistency: N/A

912 – Isolation: Required

913 – Durability: N/A

914 **Error messages:**

915

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0213	Instance not found	Mandatory	Class implem.	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

916

917 ### 6.3.2 DeleteInstance

918 **Purpose:**

919 Deletes an instance.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
InstancePath	InstancePath	Mandatory	Instance path of the instance to be deleted (Context Parameter)

Operation Output Parameters:

None.

Description:

The *DeleteInstance* operation deletes the instance referenced by *InstancePath*.

The existence of other instances may depend on the instance to be deleted. There are multiple types of dependent instances, and multiple options to handle such dependent instances, as defined in 5.8.9.

NOTE Any dependent instances that are deleted may reside in a different namespace (which may reside in a different WBEM server) than the instance referenced by *InstancePath*.

In case of error, the consistency requirements defined in DSP0004 cannot be guaranteed, but should be attempted to be satisfied in a best effort approach. Such an approach may be to delete non-dependent instances first. In case of error, only a subset of the instances to be deleted may have been deleted, but each instance shall have either been deleted completely or not at all.

The effects of the deletion of any instances on managed resources shall be defined elsewhere. For example, a management profile may define that the lifecycle of the instance is coupled with the lifecycle of some underlying managed resource, and that this resource shall be deleted when the instance is deleted.

Preconditions:

- The instance referenced by *InstancePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0213.
- The creation class of the instance referenced by *InstancePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0214.
- The namespace of the instance referenced by *InstancePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.

Postconditions:

- The instance referenced by *InstancePath* shall have been deleted.
- Any implicit deletions of dependent instances shall have happened, as defined in 5.8.9.
- Any effects of the deletion of all of these instances on any managed resources shall have happened.
- The consistency requirements defined in DSP0004 shall be satisfied for any instances related to the deleted instances.

- Requirements on ACID properties:
  - Atomicity: Required, if dependent instances are handled by rejection, as defined in 5.8.9. Recommended, if dependent instances are handled by delete propagation, as defined in 5.8.9.
  - Update Consistency: Required, if dependent instances are handled by rejection, as defined in 5.8.9. Recommended, if dependent instances are handled by delete propagation, as defined in 5.8.9.
  - Isolation: Required, if dependent instances are handled by rejection, as defined in 5.8.9. Recommended, if dependent instances are handled by delete propagation, as defined in 5.8.9.
  - Durability: Required.

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0213	Instance not found	Mandatory	Class implem.	
WIPG0246	Instance cannot be deleted due to referencing association	Optional	Class implem.	
WIPG0247	Instance cannot be deleted due to multiplicity underflow	Optional	Class implem.	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

6.3.3 ModifyInstance

Purpose:

Changes property values of a given instance.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
InstancePath	InstancePath	Mandatory	Instance path of the instance to be modified (Context Parameter)
ModifiedInstance	InstanceSpecification	Mandatory	Representation of the modified instance, specifying the new property values
IncludedProperties	PropertyName [ ]	Optional	NULL, or unordered set of property names, acting as a restricting filter on the properties to be modified

Operation Output Parameters:

None.

Description:

The *ModifyInstance* operation changes property values of the instance referenced by *InstancePath*.

The set of properties to be changed shall be determined using the following algorithm:

- Initially, the set of properties to be changed is the set of properties specified in *ModifiedInstance*.
- If the *IncludedProperties* operation input parameter is supported by the WBEM protocol and if its value is not NULL, it acts as a restricting filter on the properties to be changed such that any properties exposed by the creation class of the instance that are not named in that operation parameter are removed from the set of properties to be changed. Any duplicate or invalid property names in the *IncludedProperties* operation input parameter shall be ignored. A non-NULL empty *IncludedProperties* list removes all properties from that set.
- Any key properties and non-modifiable properties are removed from the set of properties to be changed. As a result, specifying such properties in *ModifiedInstance* or *IncludedProperties* does not cause an error.

NOTE The modifiability of properties can be defined in the schema and in management profiles.

Conformant WBEM protocols may restrict *ModifiedInstance* to specify all properties exposed by the creation class of the instance referenced by *InstancePath*.

Preconditions:

- The instance referenced by *InstancePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0213.
- The creation class of the instance referenced by *InstancePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0214.

- The namespace of the instance referenced by *InstancePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- The creation class of *ModifiedInstance* shall be the creation class of the instance referenced by *InstancePath* or a superclass of that class. If this is not satisfied, the operation shall fail, indicating WIPG0208.
- Any properties specified in *ModifiedInstance* shall be from the set of properties exposed by the creation class of *ModifiedInstance*. If this is not satisfied, the operation shall fail, indicating WIPG0208.

Postconditions:

- The values of the properties shall have been modified as defined in the Description paragraph for this operation.
- The values of key properties and non-modifiable properties shall not have been modified.
- Other properties may have changed as a result of side effects of changing properties, behavior defined in referencing specifications, or volatility of properties.
- The consistency requirements defined in DSP0004 shall be satisfied for the modified instance.
- Requirements on ACID properties:
  - Atomicity: Required
  - Update Consistency: Required
  - Isolation: Required
  - Durability: Required

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure, class implem.	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0213	Instance not found	Mandatory	Class implem.	
WIPG0220	No such property	Mandatory	Class implem.	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

6.3.4 CreateInstance

Purpose:

Creates an instance of a given class.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
ClassPath	ClassPath	Mandatory	Class path of the creation class of the instance to be created (Context Parameter)
NewInstance	InstanceSpecification	Optional	Instance representation specifying the initial property values for the instance to be created

Operation Output Parameters:

Generic Name	Generic Type	Requirement	Description
InstancePath	InstancePath	Mandatory	Instance path of the new instance

Description:

The *CreateInstance* operation creates an instance of the creation class referenced by *ClassPath* in the same namespace as that creation class and returns the instance path of the new instance.

The creation class is interpreted in a non-polymorphic way; that is, the creation class of the newly created instance shall be specified creation class (and not a subclass thereof).

The newly created instance shall have all properties exposed by the creation class referenced by *ClassPath*.

For each property, its initial value in the new instance shall be determined as follows:

- If the *NewInstance* operation input parameter is supported, and if the property is included in *NewInstance*, its value is used as the initial value. That is also the case if that value is NULL.
- Else, if an initialization constraint is defined for the property (that is, through the class-defined property default value, a use of the PropertyConstraint qualifier, or by a management profile), a value satisfying that constraint is used as the initial value.
- Else, the initial value is implementation-defined.

Key properties and non-writeable properties included in *NewInstance* shall be treated like any other properties; the creation of an instance does not have the restrictions a subsequent modification has.

Volatile properties may change their values immediately after the instance has been created.

Instance creation based upon input data other than initial property values can be done using CIM methods. For example, creation of an instance of *CIM_ComputerSystem* representing a virtual computer system could be done using a *CreateVirtualComputerSystem( )* method taking a higher-level specification of the virtual computer system as input.

Other instances may come into existence implicitly during the course of processing the *CreateInstance* operation. As defined in DSP1001, management profiles may specify the rules for such implicitly created instances.

Any such implicitly created instances may reside in the same or a different namespace (which may reside in a different WBEM server) than the namespace of the creation class referenced by *ClassPath*.

In case of error, the consistency requirements defined in DSP0004 should be attempted to be satisfied in a best effort approach. In case of error, only a subset of the instances to be created may have been created, but each instance shall have either been created completely or not at all.

As defined in DSP1001, management profiles may specify the effects of the creation of instances on managed resources. For example, a management profile may define that the lifecycle of the instance is coupled with the lifecycle of some underlying managed resource, and that this resource shall be created when the instance is created.

Preconditions:

- The instance to be created shall not exist in the namespace specified by *ClassPath*. If this is not satisfied, the operation shall fail, indicating WIPG0216.
- The class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0214.
- The namespace of the class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- The creation class of *NewInstance* shall be the class referenced by *ClassPath* or a superclass of that class. If this is not satisfied, the operation shall fail, indicating WIPG0208.
- Any properties specified in *NewInstance* shall be from the set of properties exposed by the class referenced by *ClassPath*. If this is not satisfied, the operation shall fail, indicating WIPG0208.

- If the schema definition of the class referenced by *ClassPath* or any implemented management profiles require that *NewInstance* includes a property, but that property is not included in *NewInstance*, the operation shall fail, indicating WIPG0249.
- If the schema definition of the class referenced by *ClassPath* or any implemented management profiles require that *NewInstance* does not include a property, but that property is included in *NewInstance*, the operation shall fail, indicating WIPG0249.

Postconditions:

- The instance shall have been created as defined in the Description paragraph for this operation.
- Any management profile defined implicit creations of other instances shall have happened.
- Any management profile defined effects of the creation of all of these instances on any managed resources shall have happened.
- Requirements on ACID properties:
  - Atomicity: Required
  - Update Consistency: Required
  - Isolation: Required
  - Durability: Required

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure, class implem.	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0216	Instance already exists	Mandatory	Class implem.	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

6.4 Instance enumeration operations

This subclause defines server operations that enumerate instances and return their representations and instance paths by means of subsequent pull operations.

The common pattern for these operations is that an enumeration session gets established through an "Open" operation, also establishing the kind of operation and the kind of items to be returned (instance representations together with instance paths, or just instance paths), and subsequent repeated executions of a "Pull" operation on the enumeration session are used to retrieve the items. Optionally, the "Open" operation can also pull a first set of items.

The pulled instance enumeration operations consist of the following individual operations:

- Open operations:

  OpenEnumerateInstances – Open an enumeration of instances of a given class for returning their representations and instance paths

  OpenAssociators – Open an enumeration of instances associated to a given source instance for returning their representations and instance paths

  OpenReferences – Open an enumeration of association instances referencing a given source instance for returning their representations and instance paths

  OpenQueryInstances – Open an enumeration of instances representing a query result for returning only their instance representations

- Pull operations:

  PullInstancesWithPath – Pull operation for retrieving instance representations with instance paths

  PullInstances – Pull operation for retrieving instance representations (without instance paths), representing query results

- Other operations:

  CloseEnumeration – Close an open enumeration

6.4.1 General behavioral rules

A central concept of the pulled instance enumeration operations is the "enumeration session". An enumeration session can be thought of as a context in which the operations perform their work, and which determines the set of instances to be enumerated. In order to process the operations related to an enumeration session, some of the operation parameters of the Open operation need to be maintained as long as the enumeration session is open, as well as some state data about where the enumeration session is with respect to instances already returned.

From a WBEM client's perspective, an enumeration session is represented as an enumeration context value. A successful Open operation establishes the enumeration session and returns an enumeration context value representing the open enumeration session. The enumeration context value is used as an

operation input/output parameter in subsequent Pull operations on that enumeration session. The enumeration context value shall uniquely identify the open enumeration session within the target namespace of the Open operation that established the enumeration session. This does not require the enumeration context value to be time-unique, i.e., it may be reused for a new enumeration session after the old enumeration session was closed. It is valid for a WBEM server to use NULL as an enumeration context value representing a closed enumeration session, but a WBEM client shall not rely on that to detect that an enumeration session has been closed.

Defining the enumeration context value in Pull operations not only as an operation input parameter but also as an operation output parameter allows the WBEM server to change the enumeration context value during the execution of a Pull operation. This allows for different implementation approaches for the WBEM server, which are transparent for the WBEM client.

Example approaches are:

- maintaining any state data describing the enumeration session internally in the WBEM server. In this approach, the enumeration context value does not need to change in subsequent Pull operations. It is used by the WBEM server only to identify the internal state data for the open enumeration session, but it is not used to store any of the state data in it. A variation of this approach is to hand back modified enumeration context values for additional WBEM server side sequence checking.
- maintaining any state data describing the enumeration session on the WBEM client side only. In this approach, all state data is stored in the enumeration context value, and the WBEM server does not maintain any state data about the enumeration session, essentially being completely stateless with respect to the enumeration session.
- a combination of the two previous approaches

A WBEM server may support keeping enumeration sessions open across connection terminations and shutdowns of the server. Objects may be created, deleted or modified concurrently with an enumeration session that involves these objects. Such changes may or may not be reflected in the enumeration set. Therefore, there is no guarantee to the WBEM client that the enumeration set represents a consistent snapshot of its objects at a point in time. However, the WBEM server should make a best effort attempt for the returned enumeration set to represent a consistent snapshot of its objects at a point in time. The order of objects in the enumeration set is undefined.

This specification does not define any restrictions on the number of enumeration sessions that can be established or executed on concurrently in the same WBEM server or by the same WBEM client. This remains true even if the enumeration sets of such concurrently established enumeration sessions contain the same objects.

With the exception of the CloseEnumeration operation, all operations on a particular enumeration session shall be executed sequentially. An enumeration session can be open or closed. The enumeration session is considered open if operations using its enumeration context value as an operation input parameter can be executed successfully. It is opened by the successful completion of an Open operation and closed by one of the following:

- Successful completion of a CloseEnumeration operation
- Successful completion of an Open or Pull operation that has its *EndOfSequence* operation output parameter set to TRUE. In other words, reaching the end of the enumeration set closes the enumeration session implicitly
- Unsuccessful completion of a Pull operation when *ContinueOnError* had not been requested
- WBEM server side decision to close the enumeration session based upon an operation timeout
- WBEM server side decision to close an enumeration session during an operation on that enumeration session based upon exceeding server limits

A conformant WBEM server may support closure of enumeration sessions based upon exceeding server limits. Potential examples for such a decision may be Pull operations with no objects requested that are repeated with a high frequency on the same enumeration session. If a WBEM server supports closure of enumeration sessions based upon exceeding server limits, it shall make the decision to close an enumeration session during an operation on that enumeration session. (There is no way to indicate the reason for the closure if the decision is made elsewhere.)

6.4.2 Common operation parameters for the open operations

This subclause defines commonly used operation parameters for the Open operations. The description of the individual Open operations references these operation parameters as appropriate. However, not every Open operation uses every one of these common operation parameters.

6.4.2.1 EnumerationContext

The *EnumerationContext* operation output parameter is the enumeration context value representing the enumeration session. See 6.4.1 for a definition of the concepts of *enumeration session* and *enumeration context value*.

6.4.2.2 EndOfSequence

NOTE This operation output parameter is also used for Pull operations.

The *EndOfSequence* operation output parameter indicates whether the enumeration session is exhausted.

If *EndOfSequence* is TRUE upon successful completion of an operation, no more objects are available and the WBEM server shall have closed the enumeration session, releasing any possibly allocated compute resources related to the enumeration session.

If the returned enumeration set is empty, it is valid for a WBEM server to set *EndOfSequence* to TRUE, even if *MaxObjectCount* was 0. In this case, the enumeration session will be closed upon successful completion of the operation.

If *EndOfSequence* is FALSE upon successful completion of an operation, there may be additional elements available and the WBEM server shall not have closed the enumeration session.

6.4.2.3 FilterQueryLanguage and FilterQueryString

The *FilterQueryLanguage* and *FilterQueryString* operation input parameters define a filter query that acts as an additional restricting filter on the set of instances about which information is returned.

Support for the *FilterQueryLanguage* and *FilterQueryString* operation parameters is conditional on support in the WBEM protocol for filter queries in pulled instance enumeration operations.

If the WBEM protocol supports filter queries in pulled instance enumeration operations, the following rules apply:

- Conformant WBEM protocols shall require that the DMTF Filter Query Language (FQL) defined in DSP0212 is supported for the filter queries. Conformant WBEM protocols may support additional filter query languages.
- If *FilterQueryLanguage* is not NULL, additional filtering is requested and the following rules apply:
  - *FilterQueryLanguage* shall specify a valid query language and *FilterQueryString* shall be a valid query in that query language. Neither the query language nor the format of the filter query is defined by this specification. Conformant WBEM protocols shall

define a mechanism whereby WBEM servers can declare the set of query languages that are valid for *FilterQueryLanguage*.

  - A filter query may specify any result set (e.g., SELECT list), but because the purpose of the filter query is to restrict the set of instances about which information is returned, its result set shall be ignored. The filter query shall not define any ordering criteria. The filter query shall not define any grouping of objects. Operations using filter queries may specify additional constraints on the filter query.

  - If the WBEM server infrastructure does not support filtered enumerations, the WBEM server shall return failure with message WIPG0237 (Filter queries not supported by WBEM server infrastructure).

  - If the CIM class implementation does not support filtered enumerations, the WBEM server shall return failure with message WIPG0244 (Filter queries not supported by class implementation).

- If *FilterQueryLanguage* is NULL, no additional filtering shall take place, and *FilterQueryString* shall be NULL.

  - If *FilterQueryString* is not NULL, the WBEM server shall return failure with message WIPG0208 (Invalid operation input parameter value).

If the WBEM protocol does not support filter queries in pulled instance enumeration operations, no additional filtering shall take place.

6.4.2.4 OperationTimeout

The *OperationTimeout* operation input parameter determines the "operation timeout". The operation timeout is the minimum time the WBEM server shall maintain the open enumeration session after the last Open or Pull operation (unless the enumeration session was closed during that last operation). If the operation timeout is exceeded, the WBEM server may close the enumeration session at any time, releasing any possibly allocated compute resources related to the enumeration session.

Support for the *OperationTimeout* operation parameter in a conformant WBEM protocol is mandatory.

An *OperationTimeout* of 0 means that there is no operation timeout, i.e., the enumeration session is never closed based on time.

If *OperationTimeout* is NULL, the WBEM server shall choose an operation timeout.

All other values for *OperationTimeout* specify the operation timeout in seconds.

A WBEM server may restrict the set of allowable values for *OperationTimeout*. This specifically includes the possibility for the WBEM server to not allow 0 (no timeout). If the specified value is not an allowable value, the WBEM server shall return failure with error message WIPG0242 (Invalid timeout). Conformant WBEM protocols shall define a mechanism whereby WBEM servers can declare the allowable values for *OperationTimeout*.

6.4.2.5 ContinueOnError

The *ContinueOnError* operation input parameter, if TRUE, requests continuation on error. Continuation on error is the ability to resume an enumeration session successfully after a Pull operation that returned an error. A conformant WBEM server may support continuation on error. Conformant WBEM protocols shall define a mechanism whereby WBEM servers can declare support for continuation on error.

Support for the *ContinueOnError* operation parameter is conditional on support in the WBEM protocol for client side control of continuation on error for pulled instance enumeration operations.

If the WBEM protocol supports client side control of continuation on error for pulled instance enumeration operations, the following rules apply:

- If a WBEM server does not support continuation on error and if *ContinueOnError* is TRUE, it shall return failure with error message WIPG0235 (Continuation on error not supported).
- If a WBEM server supports continuation on error, it shall support it as follows: If *ContinueOnError* is TRUE, the enumeration session shall remain open when a Pull operation returns failure, and any subsequent successful Pull operations shall return the set of elements that would have been returned if the failing Pull operations had been successful, subject to the consistency rules defined in 5.8. If *ContinueOnError* is FALSE, the enumeration session shall be closed when a Pull operation returns failure.

If the WBEM protocol does not support client side control of continuation on error for pulled instance enumeration operations, it shall define requirements for the behavior of the WBEM server with respect to continuation on error.

6.4.2.6 MaxObjectCount

NOTE This operation output parameter is also used for Pull operations.

The *MaxObjectCount* operation input parameter defines the maximum number of objects that may be returned by this operation. Any uint32 number is valid, including 0. The WBEM server may deliver any number of objects up to *MaxObjectCount* but shall not deliver more than *MaxObjectCount* objects.

Support for the *MaxObjectCount* operation parameter in a conformant WBEM protocol is mandatory.

A conformant WBEM server implementation may choose to never return any elements during an operation, regardless of the value of *MaxObjectCount*.

A WBEM client may use a *MaxObjectCount* value of 0 to specify that it does not want to retrieve any instances in the operation.

6.4.3 OpenEnumerateInstances

Purpose:

Establish and open an enumeration session for enumerating the instances of a given class and optionally return a first set of their instance representations and instance paths.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
EnumClassPath	ClassPath	Mandatory	Class path of the class whose instances are to be enumerated (Context Parameter)
FilterQueryString	QueryString	Conditional	NULL, or query string of a filter query that is acting as an additional restricting filter on the set of instances to be enumerated, as defined in 6.4.2.3 Condition: WBEM protocol supports filter queries for pulled instance enumeration operations.

Generic Name	Generic Type	Requirement	Description
FilterQueryLanguage	QueryLanguage	Conditional	NULL, or query language of the filter query specified in *FilterQueryString*, as defined in 6.4.2.3 Condition: WBEM protocol supports filter queries for pulled instance enumeration operations.
IncludedProperties	PropertyName [ ]	Optional	NULL, or unordered set of property names to be included, acting as a restricting filter on the properties included in the returned instance representations
OperationTimeout	uint32	Mandatory	Operation timeout, as defined in 6.4.2.4
ContinueOnError	boolean	Conditional	Indicates whether the enumeration session should be continued in case of error, as defined in 6.4.2.5 Condition: WBEM protocol supports client side control of continuation on error for pulled instance enumeration operations.
MaxObjectCount	uint32	Mandatory	Maximum number of instances that may be returned by this operation, as defined in 6.4.2.6

Operation Output Parameters:

Generic Name	Generic Type	Requirement	Description
InstanceList	InstanceSpecificationWithPath [ ]	Mandatory	Sequence of the returned first set of instance representations and instance paths
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.2.1
EndOfSequence	boolean	Mandatory	Indicates end of sequence for the enumeration session, as defined in 6.4.2.2

Description:

The *OpenEnumerateInstances* operation establishes and opens an enumeration session for enumerating all instances of the class referenced by *EnumClassPath*, including instances of any of its subclasses. That enumeration session allows retrieving the instance representations and instance paths of these instances through successive *PullInstancesWithPath* operations (see 6.4.8). Retrieval of a first set of instance representations and instance paths may be requested by setting *MaxObjectCount* to a value > 0.

The set of instances to be enumerated throughout the entire enumeration session shall be determined using the following algorithm:

- Initially, the set of instances to be enumerated is the set of instances in the namespace of the class referenced by *EnumClassPath*, whose creation class is the class referenced by *EnumClassPath* or a subclass of that class.

- If the WBEM protocol supports filter queries for pulled instance enumeration operations (that is, the *FilterQueryString* and *FilterQueryLanguage* operation parameters) and *FilterQueryLanguage* is not NULL, *FilterQueryString* acts as a restricting filter on the instances to be enumerated such that any instances not selected by the filter query for its result set are removed from the set of instances. The filter query shall query only the class referenced by *EnumClassPath*. See also 6.4.2.3.

The set of instances to be enumerated throughout the entire enumeration session should not contain any duplicate instances, as defined in 5.8.4. Because instances to be enumerated all exist in the same namespace, a determination of duplicate instances (for example by a WBEM client) can be done on the basis of their model paths only.

The set of instances to be returned (as instance representations and instance paths) is the first set of instances from the set of instances to be enumerated throughout the entire enumeration session, such that no more than *MaxObjectCount* instances are returned. Returning no instances does not imply that the enumeration session has been exhausted. Only the *EndOfSequence* operation output parameter indicates whether the enumeration session has been exhausted.

The set of properties to be included in any returned instance representations shall be determined using the following algorithm:

- Initially, the set of properties to be included is the set of properties exposed by the creation class of the instance. This includes all the duplicates of any duplicate non-overridden properties.
- If the *IncludedProperties* operation input parameter is supported by the WBEM protocol and if its value is not NULL, it acts as a restricting filter on the properties to be included in the returned instance representations such that any properties exposed by the creation class of the instance that are not named in that operation parameter are removed from the set of properties to be included. Any duplicate or invalid property names in the *IncludedProperties* operation input parameter shall be ignored. A non-NULL empty *IncludedProperties* list removes all properties from the set of properties to be included.
- Conformant WBEM protocols may specify rules that cause properties with a value of NULL to be removed from the set of properties to be included.

Preconditions:

- The class referenced by *EnumClassPath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0214.
- The namespace of the class referenced by *EnumClassPath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- If a filter query is specified,
  - the query language specified in the *FilterQueryLanguage* operation parameter shall be valid. If this is not satisfied, the operation shall fail, indicating WIPG0221.
  - the query specified in the *FilterQueryString* operation parameter shall be a valid query in the query language specified in the *FilterQueryLanguage* operation parameter. If this is not satisfied, the operation shall fail, indicating WIPG0222 or WIPG0223.

Postconditions:

- The enumeration session shall have been established and opened.
- A first set of instance representations and instance paths shall have been returned as described in the Description paragraph for this operation.

- Requirements on ACID properties:
  - Atomicity: Required (related to the creation of an enumeration context that is maintained by the WBEM server)
  - Update Consistency: N/A
  - Isolation: Required at the level of single instances, as defined in 5.8.
  - Durability: Required (related to creation of an enumeration context that is maintained by the WBEM server)

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0242	Invalid timeout	Mandatory	Infrastructure, class implem.	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0235	Continuation on error not supported	Mandatory	Infrastructure, class implem.	
WIPG0237	Filter queries not supported by WBEM server infrastructure	Optional	Infrastructure	
WIPG0244	Filter queries not supported by class implementation	Optional	Class implem.	
WIPG0221	Unknown query language	Mandatory	Infrastructure, class implem.	
WIPG0222	Query language feature not supported	Mandatory	Infrastructure, class implem.	
WIPG0223	Invalid query	Mandatory	Infrastructure, class implem.	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

6.4.4 OpenAssociators

Purpose:

Establish and open an enumeration session for enumerating the instances that are associated with a given source instance and optionally return a first set of their instance representations and instance paths.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
SourceInstancePath	InstancePath	Mandatory	Instance path of the source instance (Context Parameter)
AssociationClassName	ClassName	Mandatory	NULL, or name of the association class, acting as a restricting filter on the returned instances
AssociatedClassName	ClassName	Mandatory	NULL, or name of the associated class on any far end of the association, acting as a restricting filter on the returned instances
SourceRoleName	PropertyName	Mandatory	NULL, or name of the role on the source end of the association, acting as a restricting filter on the returned instances
AssociatedRoleName	PropertyName	Mandatory	NULL, or name of the role on any far end of the association, acting as a restricting filter on the returned instances
FilterQueryString	QueryString	Conditional	NULL, or query string of a filter query that is acting as an additional restricting filter on the set of returned instances, as defined in 6.4.2.3 Condition: WBEM protocol supports filter queries for pulled instance enumeration operations.
FilterQueryLanguage	QueryLanguage	Conditional	NULL, or query language of the filter query specified in *FilterQueryString*, as defined in 6.4.2.3 Condition: WBEM protocol supports filter queries for pulled instance enumeration operations.
IncludedProperties	PropertyName [ ]	Optional	NULL, or unordered set of property names to be included, acting as a restricting filter on the properties included in the returned instances
OperationTimeout	uint32	Mandatory	Operation timeout, as defined in 6.4.2.4

Generic Name	Generic Type	Requirement	Description
ContinueOnError	boolean	Conditional	Indicates whether the enumeration session should be continued in case of error, as defined in 6.4.2.5 Condition: WBEM protocol supports client side control of continuation on error for pulled instance enumeration operations.
MaxObjectCount	uint32	Mandatory	Maximum number of instances that may be returned by this operation, as defined in 6.4.2.6

Operation Output Parameters:

Generic Name	Generic Type	Requirement	Description
InstanceList	InstanceSpecificationWithPath [ ]	Mandatory	Sequence of the returned first set of instance representations and instance paths
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.2.1
EndOfSequence	boolean	Mandatory	Indicates end of sequence for the enumeration session, as defined in 6.4.2.2

Description:

The *OpenAssociators* operation establishes and opens an enumeration session for enumerating instances that are associated with the specified source instance. That enumeration session allows retrieving the instance representations and instance paths of these instances through successive *PullInstancesWithPath* operations (see 6.4.8). Retrieval of a first set of those instances together with their instance paths may be requested by setting *MaxObjectCount* to a value > 0.

The set of instances to be enumerated throughout the entire enumeration session shall be determined using the following algorithm:

- Initially, the set of instances to be enumerated is the set of all instances associated to the source instance referenced by *SourceInstancePath*. These associations may be instances of different association classes. If the source instance does not exist, the operation shall succeed with an empty result set (even when its creation class does not exist). However, if the namespace of the source instance does not exist, the operation shall fail, indicating WIPG0204.

  The result set should not contain any duplicate instances, as defined in 5.8.4. However, different far ends may reference the same instance, and in such cases, the instance shall be contained in the result set once for each such reference.

- If the *AssociationClassName* operation input parameter is not NULL, it acts as a restricting filter on the instances to be enumerated such that each instance that is associated with the source instance using an association whose creation class or one of its superclasses does not have the name specified in *AssociationClassName*, is removed from the set of instances to be enumerated. There shall be no validity checking performed for the *AssociationClassName* operation input parameter; if the specified class does not exist, the operation shall succeed with an empty result (because the filter did not match).

- If the *AssociatedClassName* operation input parameter is not NULL, it acts as a restricting filter on the instances to be enumerated such that each instance whose creation class or one of its superclasses does not have the name specified in *AssociatedClassName*, is removed from the set of instances to be enumerated. There shall be no validity checking performed for the *AssociatedClassName* operation input parameter; if the specified class does not exist, the operation shall succeed with an empty result (because the filter did not match).

  NOTE Specifying a non-NULL value for *AssociatedClassName* ensures that the returned instances have the class specified in *AssociatedClassName* as a common superclass.

- If the *SourceRoleName* operation input parameter is not NULL, it acts as a restricting filter on the instances to be enumerated such that each instance that is associated with the source instance using an association class that has a role name on the source end that is not the role name specified in *SourceRoleName*, is removed from the set of instances to be enumerated. There shall be no validity checking performed for the *SourceRoleName* operation input parameter; if the specified role does not exist, the operation shall succeed with an empty result (because the filter did not match).

- If the *AssociatedRoleName* operation input parameter is not NULL, it acts as a restricting filter on the instances to be enumerated such that each instance that is associated with the source instance using an association class that has a role name on the end referencing that instance that is not the role name specified in *AssociatedRoleName*, is removed from the set of instances to be enumerated. There shall be no validity checking performed for the *AssociatedRoleName* operation input parameter; if the specified role does not exist, the operation shall succeed with an empty result (because the filter did not match).

- If the WBEM protocol supports filter queries for pulled instance enumeration operations (that is, the *FilterQueryString* and *FilterQueryLanguage* operation parameters) and *FilterQueryLanguage* is not NULL, *FilterQueryString* acts as a restricting filter on the instances to be enumerated such that any instances not selected by the filter query for its result set are removed from the set of instances. The filter query shall query only the class specified in *AssociatedClassName* (e.g., in the CQL FROM-clause). See also 6.4.2.3.

The set of instances to be enumerated throughout the entire enumeration session should not contain any duplicate instances, as defined in 5.8.4. Because the set of returned instances contains only instances that exist in the same namespace, a determination of duplicate instances can be done on the basis of their model paths only.

The set of instances to be returned (as instance representations and instance paths) is the first set of instances from the set of instances to be enumerated throughout the entire enumeration session, such that no more than *MaxObjectCount* instances are returned. Returning no instances does not imply that the enumeration session has been exhausted. Only the *EndOfSequence* operation output parameter indicates whether the enumeration session has been exhausted.

The set of properties to be included in any returned instances shall be determined using the following algorithm:

- Initially, the set of properties to be included is the set of properties exposed by the creation class of the instance. This includes all the duplicates of any duplicate non-overridden properties.
- If the *IncludedProperties* operation input parameter is supported by the WBEM protocol and if its value is not NULL, it acts as a restricting filter on the properties to be included in the returned instances such that any properties exposed by the creation class of the instance that are not named in that operation parameter are removed from the set of properties to be included. Any duplicate or invalid property names in the *IncludedProperties* operation input parameter shall be ignored. A non-NULL empty *IncludedProperties* list removes all properties from the set of properties to be included.
- Conformant WBEM protocols may specify rules that cause properties with a value of NULL to be removed from the set of properties to be included.

Preconditions:

- The namespace of the source instance referenced by *SourceInstancePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- If a filter query is specified,
  - the query language specified in the *FilterQueryLanguage* operation parameter shall be valid. If this is not satisfied, the operation shall fail, indicating WIPG0221.
  - the query specified in the *FilterQueryString* operation parameter shall be a valid query in the query language specified in the *FilterQueryLanguage* operation parameter. If this is not satisfied, the operation shall fail, indicating WIPG0222 or WIPG0223.
  - the *AssociatedClassName* operation input parameter shall be non-NULL. If this is not satisfied, the operation shall fail, indicating WIPG0208.
- The *IncludedProperties* operation parameter, if supported by the WBEM protocol, shall only be specified with a non-NULL value if the *AssociatedClassName* operation input parameter is also non-NULL. If this is not satisfied, the operation shall fail, indicating WIPG0208.
- The namespace of any returned instance paths shall exist. If it does not exist, the operation may fail, indicating WIPG0204. Note that cross-namespace association traversals may return instance paths in a server or namespace that is different from the server or namespace of the source instance.
- The creation class of any returned instance paths shall exist in their namespace. If it does not exist, the operation may fail, indicating WIPG0214.

Postconditions:

- The enumeration session shall have been established and opened.
- A first set of instances with their instance paths shall have been returned as described in the Description paragraph for this operation.
- Requirements on ACID properties:
  - Atomicity: Required (related to the creation of an enumeration context that is maintained by the WBEM server)
  - Update Consistency: N/A

– Isolation: Required at the level of single instances, as defined in 5.8.

– Durability: Required (related to creation of an enumeration context that is maintained by the WBEM server)

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	For input namespace
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0242	Invalid timeout	Mandatory	Infrastructure, class implem.	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0235	Continuation on error not supported	Mandatory	Infrastructure, class implem.	
WIPG0237	Filter queries not supported by WBEM server infrastructure	Optional	Infrastructure	
WIPG0244	Filter queries not supported by class implementation	Optional	Class implem.	
WIPG0221	Unknown query language	Mandatory	Infrastructure, class implem.	
WIPG0222	Query language feature not supported	Mandatory	Infrastructure, class implem.	
WIPG0223	Invalid query	Mandatory	Infrastructure, class implem.	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0204	Namespace not found	Optional	Infrastructure	For namespace of returned instance paths
WIPG0214	Class not found	Optional	Infrastructure	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

6.4.5 OpenReferences

Purpose:

Establish and open an enumeration session for enumerating the association instances that reference a given source instance and optionally return a first set of their instance representations and instance paths.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
SourceInstancePath	InstancePath	Mandatory	Instance path of the source instance (Context Parameter)
AssociationClassName	ClassName	Mandatory	NULL, or name of the association class, acting as a restricting filter on the returned instances
SourceRoleName	PropertyName	Mandatory	NULL, or name of the role on the source end of the association, acting as a restricting filter on the returned instances
FilterQueryString	QueryString	Conditional	NULL, or query string of a filter query that is acting as an additional restricting filter on the set of returned instances, as defined in 6.4.2.3 Condition: WBEM protocol supports filter queries for pulled instance enumeration operations.
FilterQueryLanguage	QueryLanguage	Conditional	NULL, or query language of the filter query specified in *FilterQueryString*, as defined in 6.4.2.3 Condition: WBEM protocol supports filter queries for pulled instance enumeration operations.
IncludedProperties	PropertyName [ ]	Optional	NULL, or unordered set of property names to be included, acting as a restricting filter on the properties included in the returned instances
OperationTimeout	uint32	Mandatory	Operation timeout, as defined in 6.4.2.4

Generic Name	Generic Type	Requirement	Description
ContinueOnError	boolean	Conditional	Indicates whether the enumeration session should be continued in case of error, as defined in 6.4.2.5 Condition: WBEM protocol supports client side control of continuation on error for pulled instance enumeration operations.
MaxObjectCount	uint32	Mandatory	Maximum number of instances that may be returned by this operation, as defined in 6.4.2.6

Operation Output Parameters:

Generic Name	Generic Type	Requirement	Description
InstanceList	InstanceSpecificationWithPath [ ]	Mandatory	Sequence of the returned first set of instance representations and instance paths
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.2.1
EndOfSequence	boolean	Mandatory	Indicates end of sequence for the enumeration session, as defined in 6.4.2.2

Description:

The *OpenReferences* operation establishes and opens an enumeration session for enumerating the association instances that reference the specified source instance. That enumeration session allows retrieving the instance representations and instance paths of these instances through successive *PullInstancesWithPath* operations (see 6.4.8). Retrieval of a first set of those instances together with their instance paths may be requested by setting *MaxObjectCount* to a value > 0.

The set of instances to be enumerated throughout the entire enumeration session shall be determined using the following algorithm:

- Initially, the set of instances to be enumerated is the set of all instances referencing the source instance referenced by *SourceInstancePath*. These associations may be instances of different association classes. If the source instance does not exist, the operation shall succeed with an empty result set (even when its creation class does not exist). However, if the namespace of the source instance does not exist, the operation shall fail, indicating WIPG0204.
- If the *AssociationClassName* operation input parameter is not NULL, it acts as a restricting filter on the instances to be enumerated such that each association instance whose creation class or one of its superclasses does not have the name specified in *AssociationClassName*, is removed from the set of instances to be enumerated. There shall be no validity checking performed for the *AssociationClassName* operation input parameter; if the specified class does not exist, the operation shall succeed with an empty result (because the filter did not match).

  NOTE Specifying a non-NULL value for *AssociationClassName* ensures that the returned instances have the class specified in *AssociationClassName* as a common superclass.

- If the *SourceRoleName* operation input parameter is not NULL, it acts as a restricting filter on the instances to be enumerated such that each association instance whose creation class does not have the role name specified in *SourceRoleName* on the end referencing the source instance, is removed from the set of instances to be enumerated. There shall be no validity checking performed for the *SourceRoleName* operation input parameter; if the specified role does not exist, the operation shall succeed with an empty result (because the filter did not match).
- If the WBEM protocol supports filter queries for pulled instance enumeration operations (that is, the *FilterQueryString* and *FilterQueryLanguage* operation parameters) and *FilterQueryLanguage* is not NULL, *FilterQueryString* acts as a restricting filter on the instances to be enumerated such that any instances not selected by the filter query for its result set are removed from the set of instances. The filter query shall query only the class specified in *AssociationClassName* (e.g., in the CQL FROM-clause). See also 6.4.2.3.

The set of instances to be enumerated throughout the entire enumeration session should not contain any duplicate instances, as defined in 5.8.4. Because the set of returned instances contains only instances that exist in the same namespace, so any determination of duplicate instances (for example by a WBEM client) may be done on the basis of their model paths.

The set of instances to be returned (as instance representations and instance paths) is the first set of instances from the set of instances to be enumerated throughout the entire enumeration session, such that no more than *MaxObjectCount* instances are returned. Returning no instances does not imply that the enumeration session has been exhausted. Only the *EndOfSequence* operation output parameter indicates whether the enumeration session has been exhausted.

The set of properties to be included in any returned instances shall be determined using the following algorithm:

- Initially, the set of properties to be included is the set of properties exposed by the creation class of the instance. This includes all the duplicates of any duplicate non-overridden properties.
- If the *IncludedProperties* operation input parameter is supported by the WBEM protocol and if its value is not NULL, it acts as a restricting filter on the properties to be included in the returned instances such that any properties exposed by the creation class of the instance that are not named in that operation parameter are removed from the set of properties to be included. Any duplicate or invalid property names in the *IncludedProperties* operation input parameter shall be ignored. A non-NULL empty *IncludedProperties* list removes all properties from the set of properties to be included.
- Conformant WBEM protocols may specify rules that cause properties with a value of NULL to be removed from the set of properties to be included.

Preconditions:

- The namespace of the source instance referenced by *SourceInstancePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- If a filter query is specified,
  - the query language specified in the *FilterQueryLanguage* operation parameter shall be valid. If this is not satisfied, the operation shall fail, indicating WIPG0221.
  - the query specified in the *FilterQueryString* operation parameter shall be a valid query in the query language specified in the *FilterQueryLanguage* operation parameter. If this is not satisfied, the operation shall fail, indicating WIPG0222 or WIPG0223.
  - the *AssociationClassName* operation input parameter shall be non-NULL. If this is not satisfied, the operation shall fail, indicating WIPG0208.

- The *IncludedProperties* operation parameter, if supported by the WBEM protocol, shall only be specified with a non-NULL value if the *AssociationClassName* operation input parameter is also non-NULL. If this is not satisfied, the operation shall fail, indicating WIPG0208.
- The namespace of any returned instance paths shall exist. If it does not exist, the operation may fail, indicating WIPG0204. Note that cross-namespace association traversals may return instance paths in a server or namespace that is different from the server or namespace of the source instance.
- The creation class of any returned instance paths shall exist in their namespace. If it does not exist, the operation may fail, indicating WIPG0214.

Postconditions:

- The enumeration session shall have been established and opened.
- A first set of instances with their instance paths shall have been returned as described in the Description paragraph for this operation.
- Requirements on ACID properties:
  - Atomicity: Required (related to the creation of an enumeration context that is maintained by the WBEM server)
  - Update Consistency: N/A
  - Isolation: Required at the level of single instances, as defined in 5.8.
  - Durability: Required (related to creation of an enumeration context that is maintained by the WBEM server)

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	For input namespace
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0242	Invalid timeout	Mandatory	Infrastructure, class implem.	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0235	Continuation on error not supported	Mandatory	Infrastructure, class implem.	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0237	Filter queries not supported by WBEM server infrastructure	Optional	Infrastructure	
WIPG0244	Filter queries not supported by class implementation	Optional	Class implem.	
WIPG0221	Unknown query language	Mandatory	Infrastructure, class implem.	
WIPG0222	Query language feature not supported	Mandatory	Infrastructure, class implem.	
WIPG0223	Invalid query	Mandatory	Infrastructure, class implem.	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0204	Namespace not found	Optional	Infrastructure	For namespace of returned instance paths
WIPG0214	Class not found	Optional	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

6.4.6 OpenQueryInstances

Purpose:

Establish and open an enumeration session for enumerating the instances of a query result in a given namespace and optionally return a first set of their instance representations.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
NamespacePath	NamespacePath	Mandatory	Namespace path of the namespace in which the query is executed (Context Parameter)
QueryString	QueryString	Mandatory	Query string of a query that defines the set of instances to be returned
QueryLanguage	QueryLanguage	Mandatory	Query language of the query specified in *QueryString*
ReturnQueryResultClass	boolean	Mandatory	Indicates whether a class definition of the query result should be returned in *QueryResultClass*
OperationTimeout	uint32	Mandatory	Operation timeout, as defined in 6.4.2.4

Generic Name	Generic Type	Requirement	Description
ContinueOnError	boolean	Conditional	Indicates whether the enumeration session should be continued in case of error, as defined in 6.4.2.5 Condition: WBEM protocol supports client side control of continuation on error for pulled instance enumeration operations.
MaxObjectCount	uint32	Mandatory	Maximum number of instances that may be returned by this operation, as defined in 6.4.2.6

1592

1593 **Operation Output Parameters:**

1594

Generic Name	Generic Type	Requirement	Description
InstanceList	InstanceSpecification [ ]	Mandatory	Sequence of the returned first set of instance representations
QueryResultClass	ClassSpecification	Mandatory	Representation of a class definition for the query result
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.2.1
EndOfSequence	boolean	Mandatory	Indicates end of sequence for the enumeration session, as defined in 6.4.2.2

1595

1596 **Description:**

1597 The *OpenQueryInstances* operation establishes and opens an enumeration session for enumerating
1598 the instances representing the result of the query specified in *QueryString* in the namespace
1599 referenced by *NamespacePath*. That enumeration session allows retrieving representations of these
1600 result instances through successive *PullInstances* operations (see 6.4.9). Retrieval of a first set of
1601 those instances may be requested by setting *MaxObjectCount* to a value > 0.

1602 The set of instances to be returned (as instance representations) is the first set of instances from the
1603 set of instances to be enumerated throughout the entire enumeration session, such that no more
1604 than *MaxObjectCount* instances are returned. Returning no instances in the *InstanceList* operation
1605 parameter does not imply that the enumeration session has been exhausted. Only the
1606 *EndOfSequence* operation output parameter indicates whether the enumeration session has been
1607 exhausted.

1608 The returned instance representations have no corresponding addressable instances that exist.

1609 If *QueryLanguage* is not NULL, it shall specify a valid query language and *QueryString* shall be a
1610 valid query in that query language. Neither the query language nor the format of the filter query is
1611 defined by this specification. Conformant WBEM protocols shall specify a mechanism for determining
1612 the set of query languages that are valid for *QueryLanguage*. The simplest way to do this is to list the
1613 set of valid query languages.

1614 The value of the *ReturnQueryResultClass* operation input parameter controls whether or not a class
1615 definition is returned in the *QueryResultClass* operation output parameter. If FALSE, then
1616 *QueryResultClass* shall be NULL. If TRUE, then the value of *QueryResultClass* shall be a class
1617 definition that defines the properties of each instance of the query result. The name of this class shall

be CIM_QueryResult. This class is only a representation of a class that has no corresponding addressable class residing in the WBEM server.

Preconditions:

- The namespace referenced by *NamespacePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- The query language specified in the *QueryLanguage* operation parameter shall be a valid query language. If this is not satisfied, the operation shall fail, indicating WIPG0221.
- The query specified in the *QueryString* operation parameter shall be a valid query in the query language specified in the *QueryLanguage* operation parameter. If this is not satisfied, the operation shall fail, indicating WIPG0222 or WIPG0223.

Postconditions:

- The enumeration session shall have been established and opened.
- A first set of instances shall have been returned as described in the Description paragraph for this operation.
- Requirements on ACID properties:
  - Atomicity: Required (related to the creation of an enumeration context that is maintained by the WBEM server)
  - Update Consistency: N/A
  - Isolation: Required at the level of single instances, as defined in 5.8.
  - Durability: Required (related to creation of an enumeration context that is maintained by the WBEM server)

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0242	Invalid timeout	Mandatory	Infrastructure, class implem.	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0235	Continuation on error not supported	Mandatory	Infrastructure, class implem.	
WIPG0221	Unknown query language	Mandatory	Infrastructure, class implem.	
WIPG0222	Query language feature not supported	Mandatory	Infrastructure, class implem.	
WIPG0223	Invalid query	Mandatory	Infrastructure, class implem.	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

6.4.7 Common operation parameters for the pull operations

This subclause defines commonly used operation parameters for the Pull operations. The description of the individual Pull operations references these operation parameters as appropriate. However, not every Pull operation uses every one of these common operation parameters.

6.4.7.1 NamespacePath

The *NamespacePath* operation input parameter references the namespace identified by the context parameter of the Open operation that established and opened the enumeration session.

6.4.7.2 EnumerationContext

The *EnumerationContext* operation input/output parameter is the enumeration context value representing the enumeration session to be used.

Support for the *EnumerationContext* operation parameter in a conformant WBEM protocol is mandatory.

When invoking the Pull operation, the enumeration session represented by *EnumerationContext* shall be open. The enumeration session shall have been established using one of the Open operations whose type of enumerated element matches the Pull operation. For the first Pull operation on an enumeration session, the value of *EnumerationContext* shall be the enumeration context value returned by a successful Open operation that established and opened that enumeration session. For any subsequent Pull operations on that enumeration session, the value of *EnumerationContext* shall be the value of *EnumerationContext* as returned by the previous Pull operation on the same enumeration session.

After completing the Pull operation, the enumeration session represented by *EnumerationContext* shall be open or closed.

6.4.7.3 EndOfSequence

The *EndOfSequence* operation output parameter when used in Pull operations behaves as defined in 6.4.2.2

6.4.7.4 MaxObjectCount

The *MaxObjectCount* operation input parameter when used in Pull operations behaves as defined in 6.4.2.6.

6.4.8 PullInstancesWithPath

Purpose:

Retrieve the next set of instance representations and instance paths from an open enumeration session.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
NamespacePath	NamespacePath	Mandatory	Namespace path of the namespace for the enumeration, as defined in 6.4.7.1 (Context Parameter)
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.7.2
MaxObjectCount	uint32	Mandatory	Maximum number of instances that may be returned by this operation, as defined in 6.4.7.4

Operation Output Parameters:

Generic Name	Generic Type	Requirement	Description
InstanceList	InstanceSpecificationWithPath [ ]	Mandatory	Next set of returned instance representations and instance paths
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.7.2
EndOfSequence	boolean	Mandatory	Indicates end of sequence for the enumeration session, as defined in 6.4.7.3

Description:

The *PullInstancesWithPath* operation retrieves the next set of instance representations and instance paths from an open enumeration session.

The enumeration session shall have been established using one of the following operations:

- OpenEnumerateInstances
- OpenAssociators
- OpenReferences

The set of instances to be returned (as instance representations and instance paths) is the next set of instances from the set of instances to be enumerated throughout the entire enumeration session, such that no more than *MaxObjectCount* instances are returned. Returning no instances does not

imply that the enumeration session has been exhausted. Only the *EndOfSequence* operation output parameter indicates whether the enumeration session has been exhausted.

The set of properties to be included in any retrieved instances shall be the as determined using the Open operation that established the enumeration session.

Preconditions:

- The namespace referenced by *NamespacePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- The enumeration session identified by *EnumerationContext* shall be open. If this is not satisfied, the operation shall fail, indicating WIPG0241.
- The value of *EnumerationContext* shall be the enumeration context value returned by the previous Open or Pull operation on the same enumeration session. If this is not satisfied, the operation shall fail, indicating WIPG0241.
- The namespace of any returned instance paths shall exist. If it does not exist, the operation may fail, indicating WIPG0204. Note that cross-namespace association traversals may return instance paths in a server or namespace that is different from the server or namespace of the source instance.
- The creation class of any returned instance paths shall exist in their namespace. If it does not exist, the operation may fail, indicating WIPG0214.

Postconditions:

- The set of instances with their instance paths shall have been returned as described in the Description paragraph for this operation.
- Requirements on ACID properties:
  - Atomicity: Required (related to updates to an enumeration context that is maintained by the WBEM server)
  - Update Consistency: N/A
  - Isolation: Required at the level of single instances, as defined in 5.8.
  - Durability: Required (related to updates to an enumeration context that is maintained by the WBEM server)

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0241	Invalid enumeration context	Mandatory	Class implem.	
WIPG0238	Pull operation has been abandoned due to enumeration context closure	Mandatory	Class implem.	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

6.4.9 PullInstances

Purpose:

Retrieve the next set of instances from an open enumeration session.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
NamespacePath	NamespacePath	Mandatory	Namespace path of the namespace for the enumeration, as defined in 6.4.7.1 (Context Parameter)
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.7.2
MaxObjectCount	uint32	Mandatory	Maximum number of instances that may be returned by this operation, as defined in 6.4.7.4

Operation Output Parameters:

Generic Name	Generic Type	Requirement	Description
InstanceList	InstanceSpecification [ ]	Mandatory	Next set of returned instance representations
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.7.2
EndOfSequence	boolean	Mandatory	Indicates end of sequence for the enumeration session, as defined in 6.4.7.3

Description:

The *PullInstances* operation retrieves the next set of instance representations without their instance paths from an open enumeration session.

The enumeration session shall have been established using one of the following operations:

- *OpenQueryInstances*

The set of instances to be returned (as instance representations) is the next set of instances from the set of instances to be enumerated throughout the entire enumeration session, such that no more than *MaxObjectCount* instances are returned. Returning no instances does not imply that the enumeration session has been exhausted. Only the *EndOfSequence* operation output parameter indicates whether the enumeration session has been exhausted.

The set of properties to be included in any retrieved instances shall be the as determined using the Open operation that established the enumeration session.

Preconditions:

- The namespace referenced by *NamespacePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- The enumeration session identified by *EnumerationContext* shall be open. If this is not satisfied, the operation shall fail, indicating WIPG0241.
- The value of *EnumerationContext* shall be the enumeration context value returned by the previous Open or Pull operation on the same enumeration session. If this is not satisfied, the operation shall fail, indicating WIPG0241.

Postconditions:

- The set of instances shall have been returned as described in the Description paragraph for this operation.
- Requirements on ACID properties:
  - Atomicity: Required (related to updates to an enumeration context that is maintained by the WBEM server)
  - Update Consistency: N/A
  - Isolation: Required at the level of single instances, as defined in 5.8.
  - Durability: Required (related to updates to an enumeration context that is maintained by the WBEM server)

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0241	Invalid enumeration context	Mandatory	Class implem.	
WIPG0238	Pull operation has been abandoned due to enumeration context closure	Mandatory	Class implem.	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

6.4.10 CloseEnumeration

Purpose:

Close an open enumeration session.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
NamespacePath	NamespacePath	Mandatory	Namespace path of the namespace for the enumeration, as defined in 6.4.7.1 (Context Parameter)
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.7.2

Operation Output Parameters:

None.

Description:

The *CloseEnumeration* operation closes the open enumeration session identified by *EnumerationContext*.

The enumeration session shall have been established using any of the Open operations.

Enumeration sessions are closed implicitly when exhausted, so this operation only needs to be used when terminating an enumeration sequence before it is exhausted.

Preconditions:

- The namespace referenced by *NamespacePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- The enumeration session identified by *EnumerationContext* shall be open. If this is not satisfied, the operation shall fail, indicating WIPG0241.
- The value of *EnumerationContext* shall be the enumeration context value returned by the previous Open or Pull operation on the same enumeration session. If this is not satisfied, the operation shall fail, indicating WIPG0241.

Postconditions:

- The enumeration session identified by *EnumerationContext* is closed.
- Requirements on ACID properties:
  - Atomicity: Required (related to updates to or deletion of an enumeration context that is maintained by the WBEM server)
  - Update Consistency: N/A
  - Isolation: Required
  - Durability: Required (related to updates to or deletion of an enumeration context that is maintained by the WBEM server)

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0241	Invalid enumeration context	Mandatory	Class implem.	
WIPG0239	Pull operation cannot be abandoned	Mandatory	Class implem.	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

6.5 Method invocation operations

This subclause defines server operations for the invocation of CIM methods.

6.5.1 InvokeMethod

Purpose:

Invoke a non-static method on an instance.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
InstancePath	InstancePath	Mandatory	Instance path of the instance the method is invoked on (Context Parameter)
MethodName	MethodName	Mandatory	Name of the method being invoked
InParmValues	ParameterValue [ ]	Mandatory	Unordered set of named input parameter values of the method

Operation Output Parameters:

Generic Name	Generic Type	Requirement	Description
OutParmValues	ParameterValue [ ]	Mandatory	Unordered set of named output parameter values of the method
ReturnValue	ReturnValue	Mandatory	Return value of the method

Description:

Invoke a CIM method using an instance path. The method may be static or non-static.

Conformant WBEM protocols shall define a mapping for the invocation of CIM methods using an instance path, including a mapping of the operation parameters defined in the tables above. These rules may map the method invocation to a single operation, map each method to its own separate operation, or define any other appropriate mapping.

If the implementation of the method could be invoked, the operation is considered successful, regardless of what the semantics of any return values or output parameters is. For example, if a method defines that a particular return value indicates an error condition, the method invocation was still successful from a perspective of the invocation operation.

Preconditions:

- The instance referenced by *InstancePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0213.
- The creation class of the instance referenced by *InstancePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0214.
- The namespace of the instance referenced by *InstancePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- The method to be invoked shall be exposed by the creation class of the instance referenced by *InstancePath*. If this is not satisfied, the operation shall fail, indicating WIPG0218.

Postconditions:

- The CIM method shall have been invoked.
- Requirements on ACID properties:
  - Atomicity: Recommended
  - Update Consistency: Recommended
  - Isolation: Recommended
  - Durability: Required

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0229	Method invocation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0218	No such method	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0213	Instance not found	Mandatory	Class implem.	
WIPG0219	Method not supported by class implementation	Mandatory	Class implem.	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

6.5.2 InvokeStaticMethod

Purpose:

Invoke a static method on a class.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
ClassPath	ClassPath	Mandatory	Class path of the class the method is invoked on (Context Parameter)
MethodName	MethodName	Mandatory	Name of the method being invoked
InParmValues	ParameterValue [ ]	Mandatory	Unordered set of named input parameter values of the method

Operation Output Parameters:

Generic Name	Generic Type	Requirement	Description
OutParmValues	ParameterValue [ ]	Mandatory	Unordered set of named output parameter values of the method
ReturnValue	ReturnValue	Mandatory	Return value of the method

Description:

Invoke a static CIM method using a class path.

Conformant WBEM protocols shall define a mapping for the invocation of CIM methods using a class path, including a mapping of the operation parameters defined in the tables above. These rules may map the method invocation to a single operation, map each method to its own separate operation, or define any other appropriate mapping.

If the implementation of the method could be invoked, the operation is considered successful, regardless of what the semantics of any return values or output parameters is. For example, if a method defines that a particular return value indicates an error condition, the method invocation was still successful from a perspective of the invocation operation.

Preconditions:

- The class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0214.
- The namespace of the class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- The method to be invoked shall be exposed by the creation class of the instance referenced by *InstancePath*. If this is not satisfied, the operation shall fail, indicating WIPG0218.

Postconditions:

- The CIM method shall have been invoked.
- Requirements on ACID properties:
  - Atomicity: Recommended
  - Update Consistency: Recommended
  - Isolation: Recommended
  - Durability: Required

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0229	Method invocation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0218	No such method	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0214	Class not found	Mandatory	Class implem.	
WIPG0219	Method not supported by class implementation	Mandatory	Class implem.	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

6.6 Class operations

This subclause defines server operations that target a single class or create a class. These operations include dealing with qualifier values defined on classes and their elements.

6.6.1 GetClass

Purpose:

Retrieve a class.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
ClassPath	ClassPath	Mandatory	Class path of the class to be retrieved (Context Parameter)
IncludeInheritedElements	boolean	Optional	Indicates whether any elements inherited from superclasses are to be included in the returned class
IncludeQualifiers	boolean	Mandatory	Indicates whether qualifier values on any returned CIM elements are to be included, as defined in 6.2.1

Operation Output Parameters:

Generic Name	Generic Type	Requirement	Description
Class	ClassSpecification	Mandatory	Retrieved class representation

Description:

The *GetClass* operation retrieves a representation of the class referenced by *ClassPath*.

The set of properties to be included in the retrieved class shall be determined using the following algorithm:

- Initially, the set of properties to be included is the set of properties exposed by the class to be retrieved. This includes all the duplicates of any duplicate non-overridden properties.
- If *IncludeInheritedElements* is FALSE, it acts as a restricting filter on the elements (properties, methods, qualifiers) to be included in the returned class such that any elements inherited into the class to be retrieved are removed from the set of properties to be included. This is also known as reducing the elements to *local-only* elements.

Preconditions:

- The class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0214.
- The namespace of the class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.

Postconditions:

- The class representation shall have been returned as defined in the Description paragraph for this operation.
- Requirements on ACID properties:
  - Atomicity: N/A
  - Update Consistency: N/A
  - Isolation: Required
  - Durability: N/A

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

6.6.2 DeleteClass

Purpose:

Delete a given class.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
ClassPath	ClassPath	Mandatory	Class path of the class to be deleted (Context Parameter)
DeleteDependents	Boolean	Optional	**EXPERIMENTAL:** Indicates whether dependent classes and instances are to be deleted as well

Operation Output Parameters:

None.

Description:

The *DeleteClass* operation deletes the class referenced by *ClassPath*.

EXPERIMENTAL

If the WBEM protocol supports the *DeleteDependents* operation parameter, the following rules apply:

- If *DeleteDependents* is TRUE, any classes that depend on the class referenced by *ClassPath* in the way described below shall be deleted, and any instances of the class referenced by *ClassPath* and of any classes depending on it shall be deleted according to the rules defined for the
- DeleteInstance operation. If these rules cause the rejection of an instance deletion, the
- DeleteClass operation shall fail.
- If *DeleteDependents* is FALSE, the *DeleteClass* operation shall fail if any classes exist that depend on the class referenced by *ClassPath* in the way described below, or if the class referenced by *ClassPath* has any instances.

EXPERIMENTAL

If the WBEM protocol does not support the *DeleteDependents* operation parameter, the *DeleteClass* operation shall fail if any classes exist that depend on the class referenced by *ClassPath* in the way described below, or if the class referenced by *ClassPath* has any instances.

For the purpose of the *DeleteClass* operation, the following classes are considered depending on the class referenced by *ClassPath*:

- Any subclasses of any class depending on the class referenced by *ClassPath*.
- Any association classes referencing any class depending on the class referenced by *ClassPath*.
- Any classes defining a method with a parameter or a return value that is
  - a reference to any class depending on the class referenced by *ClassPath*, or
  - an embedded instance of any class depending on the class referenced by *ClassPath*, or
  - an embedded class depending on the class referenced by *ClassPath*.
- Any classes defining a property that is
  - an embedded instance of any class depending on the class referenced by *ClassPath*, or
  - an embedded class depending on the class referenced by *ClassPath*.

Any classes or instances that are automatically deleted may reside in a different namespace (which may reside in a different WBEM server) than the class referenced by *ClassPath*.

In case of error, the consistency requirements defined in DSP0004 cannot be guaranteed, but should be attempted to be satisfied in a best effort approach. In case of error, only a subset of the elements to be deleted may have been deleted, but each element shall have either been deleted completely or not at all. Also, classes shall only be deleted if all of its instances could be deleted successfully.

NOTE In a non-transactional implementation, this requires an order of deletion that starts with those elements that do not depend on the deletion of other elements.

Preconditions:

- The class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0214.
- The namespace of the class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.

Postconditions:

- The class referenced by *ClassPath* shall have been deleted.
- If *DeleteDependents* was TRUE:
  - any dependent classes and instances shall have been deleted as defined in the Description paragraph for this operation, and
  - any management profile defined implicit deletions of other instances shall have happened, and
  - any management profile defined effects of the deletion of all of these instances on any managed resources shall have happened.
- The consistency requirements defined in DSP0004 shall be satisfied for any classes and instances related to the deleted classes and instances.
- Requirements on ACID properties:
  - Atomicity: Required, if dependent classes and instances are handled by rejection, as defined in 5.8.9. Recommended, if dependent classes and instances are handled by delete propagation, as defined in 5.8.9.
  - Update Consistency: Required, if dependent classes and instances are handled by rejection, as defined in 5.8.9. Recommended, if dependent classes and instances are handled by delete propagation, as defined in 5.8.9.
  - Isolation: Required, if dependent classes and instances are handled by rejection, as defined in 5.8.9. Recommended, if dependent classes and instances are handled by delete propagation, as defined in 5.8.9.
  - Durability: Required

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0224	Class has subclasses	Mandatory	Infrastructure	
WIPG0225	Class has instances	Mandatory	Infrastructure, class implem.	
WIPG0230	Class has referencing association classes	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

6.6.3 ModifyClass

Purpose:

Change a given class.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
ClassPath	ClassPath	Mandatory	Class path of the class to be changed. (Context Parameter)
ModifiedClass	ClassSpecification	Mandatory	Class representation specifying the new class definition

Operation Output Parameters:

None.

Description:

The *ModifyClass* operation changes the definition of the class referenced by *ClassPath*.

Within the restrictions specified in the preconditions, the definition of the class referenced by *ClassPath* is replaced with the definition specified in *ModifiedClass*, as follows:

- Any elements previously defined in the class to be changed (including overriding elements) that are not specified in *ModifiedClass* shall be removed from the class to be changed.
- Any elements previously defined in the class to be changed (including overriding elements) that are also specified in *ModifiedClass* shall be replaced with the definition from *ModifiedClass*.
- Any elements not previously defined in the class to be changed (including overriding elements) that are specified in *ModifiedClass* shall be added to the class to be changed, as defined in *ModifiedClass*.

Any instances whose creation class is the class referenced by *ClassPath* or one of its subclasses shall be changed to reflect the changes to the class, as follows:

- Added properties are reflected using the rules defined in the *ModifyInstance* operation when processing a list of these new properties with their values set to their class defined default values, or NULL where no class defined default value is defined.

Any other changes to the class that are compatible with the preconditions do not affect existing instances, for the following reasons:

- A compatible removal of properties from a class can only happen for overridden properties or for properties that move to a superclass, both of which is equivalent to potential changes of qualifier values and the default property value. Changes of qualifier values do not affect instances. A changed default value only affects new instances, but not existing instances.
- A compatible change of existing property definitions can only include potential changes of qualifier values and the default property value. Changes of qualifier values do not affect instances. A changed default value only affects new instances, but not existing instances.
- A compatible change of values of class qualifiers does not affect instances of the class.
- A compatible change to a method definition does not affect instances of the class.

Preconditions:

- The class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0214.
- The namespace of the class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- The name of the class defined by *ModifiedClass* shall be the name of the class referenced by *ClassPath*. If this is not satisfied, the operation shall fail, indicating WIPG0208.
- If the class referenced by *ClassPath* has a superclass, the class defined by *ModifiedClass* shall specify a superclass with the same name as that superclass. If the class referenced by *ClassPath* has no superclass, the class defined by *ModifiedClass* shall not specify a superclass. If this is not satisfied, the operation shall fail, indicating WIPG0226.
- The class defined by *ModifiedClass* shall only specify elements that when applied to the class to be modified, result in a class definition that satisfies any consistency and backward compatibility requirements defined in DSP0004. For example, qualifiers with flavor *DisableOverride* shall not be overridden, or data types of overridden properties shall not be changed. If this is not satisfied, the operation shall fail, indicating WIPG0231.

Postconditions:

- The definition of the class referenced by *ClassPath* shall have been modified as defined in the Description paragraph for this operation.
- Any instances of the class or its subclasses shall have been changed as defined in the Description paragraph for this operation.
- The consistency and backward compatibility requirements defined in DSP0004 shall be satisfied for the modified class.
- Requirements on ACID properties:
  - Atomicity: Required
  - Update Consistency: Required
  - Isolation: Required
  - Durability: Required

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0226	Superclass not found	Mandatory	Infrastructure	
WIPG0231	Incompatible class modification	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

6.6.4 CreateClass

Purpose:

Create a class.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
NamespacePath	NamespacePath	Mandatory	Namespace path of the namespace the class is to be created in (Context Parameter)
NewClass	ClassSpecification	Mandatory	Representation of the class to be created

Operation Output Parameters:

Generic Name	Generic Type	Requirement	Description
ClassPath	ClassPath	Mandatory	Class path of the new class

Description:

The *CreateClass* operation creates a class in the namespace referenced by *NamespacePath*, using the class representation in *NewClass*, and returns the class path of the new class.

If properties or methods defined in *NewClass* are intended to override properties or methods defined in a superclass of NewClass, then they shall define an *OVERRIDE* qualifier in their definition in *NewClass*. The *CreateClass* operation shall not add such qualifiers automatically.

Preconditions:

- The namespace referenced by *NamespacePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- The class to be created shall not exist in the namespace referenced by *NamespacePath*. If this is not satisfied, the operation shall fail, indicating WIPG0217.
- If *NewClass* specifies a superclass, that superclass shall exist in the namespace referenced by *NamespacePath*. If this is not satisfied, the operation shall fail, indicating WIPG0226.

  NOTE DSP0004 does not provide for inheritance relationships that cross namespace boundaries.
- The definition of *NewClass* shall satisfy any consistency requirements defined in DSP0004. If this is not satisfied, the operation shall fail, indicating WIPG0208.

Postconditions:

- The class shall have been created as defined in the Description paragraph for this operation.
- Requirements on ACID properties:
  - Atomicity: Required
  - Update Consistency: Required
  - Isolation: Required
  - Durability: Required

2086 **Error messages:**

2087

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0217	Class already exists	Mandatory	Infrastructure	
WIPG0226	Superclass not found	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

2088

2089 6.7 Class enumeration operations

2090 This subclause defines server operations that enumerate classes and return their representations and
2091 class paths.

2092 6.7.1 EnumerateClasses

2093 **Purpose:**

2094 Enumerate classes in a namespace and return these classes together with their class paths.

2095 **Operation Input Parameters:**

2096

Generic Name	Generic Type	Requirement	Description
NamespacePath	NamespacePath	Mandatory	Namespace path of the namespace the enumeration is executed on (Context Parameter)
ClassName	ClassName	Mandatory	Optional: Name of the CIM class whose subclasses are to be enumerated. If not specified, top classes are enumerated.
IncludeSubclasses	boolean	Mandatory	Indicates whether the entire tree of subclasses is to be included in the result set, in addition

Generic Name	Generic Type	Requirement	Description
IncludeInheritedElements	boolean	Mandatory	Indicates whether any elements inherited from superclasses of ClassName are to be included in the returned classes
IncludeQualifiers	boolean	Mandatory	Indicates whether qualifier values on any returned CIM elements are to be included, as defined in 6.2.1

2097

2098 **Operation Output Parameters:**

2099

Generic Name	Generic Type	Requirement	Description
ClassList	ClassSpecificationWithPath [ ]	Mandatory	Sequence of the enumerated classes with their class paths

2100

2101 **Description:**

2102 The *EnumerateClasses* operation enumerates classes (including association and indication classes)
2103 in the namespace specified in *NamespacePath* and returns their representations and class paths.

2104 *ClassName* and *IncludeSubclasses* together determine the set of classes in the result set. The set of
2105 classes in the result set is determined using the following algorithm:

2106 1) *ClassName* is optional to be specified by the WBEM client (Note that *ClassName* is
2107 mandatory to be supported by the WBEM protocol). If *ClassName* is not specified, the
2108 result set initially contains all top classes (that is, classes that do not have a superclass) in
2109 the namespace. If *ClassName* is specified, the result set initially contains the subclasses of
2110 the class specified in *ClassName* (not including the class specified in *ClassName*).

2111 2) If *IncludeSubclasses* is TRUE, then all direct and indirect subclasses of the classes that
2112 are so far in the result set are added to the result set. Otherwise, the result set is not
2113 changed.

2114 If *IncludeInheritedElements* is TRUE, then the set of CIM elements in each returned class shall
2115 consist of all elements exposed by that class. Otherwise, the set of CIM elements in each returned
2116 class shall consist only of all elements defined in the class specified in *ClassName* (including
2117 overriding elements).

2118 The consistency model defined in 5.8 applies.

2119 **Preconditions:**

2120 • The namespace referenced by *NamespacePath* shall exist. If it does not exist, the operation
2121 shall fail, indicating WIPG0204.

2122 • If *ClassName* is specified, the specified CIM class shall exist in the namespace referenced by
2123 *NamespacePath*. If this is not satisfied, the operation shall fail, indicating WIPG0214.

Postconditions:

- The enumerated classes with their class paths shall have been returned as defined in the Description paragraph for this operation.
- Requirements on ACID properties:
  - Atomicity: N/A
  - Update Consistency: N/A
  - Isolation: Required at the level of single classes, as defined in 5.8.
  - Durability: N/A

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

6.7.2 AssociatorClasses

Purpose:

Enumerate the classes that are associated with a given source class and return their class representations and class paths.

2139 **Operation Input Parameters:**

2140

Generic Name	Generic Type	Requirement	Description
ClassPath	ClassPath	Mandatory	Class path of the source class (Context Parameter)
AssociationClassName	ClassName	Mandatory	NULL, or name of the association class, acting as a restricting filter on the associated classes
AssociatedClassName	ClassName	Mandatory	NULL, or name of the associated class on any far end of the association, acting as a restricting filter on the associated classes
SourceRoleName	PropertyName	Mandatory	NULL, or name of the role on the starting end of the association, acting as a restricting filter on the associated classes
AssociatedRoleName	PropertyName	Mandatory	NULL, or name of the role on any far end of the association, acting as a restricting filter on the associated classes
IncludeQualifiers	boolean	Mandatory	Indicates whether qualifier values on any returned CIM elements are to be included, as defined in 6.2.1

2141

2142 **Operation Output Parameters:**

2143

Generic Name	Generic Type	Requirement	Description
ClassList	ClassSpecificationWithPath [ ]	Mandatory	Sequence of the returned class representations and class paths

2144

2145 **Description:**

2146 The *AssociatorClasses* operation traverses an association from a given source class on a starting
2147 end to classes on all of its far ends and returns the associated classes together with their class
2148 paths.

2149 The set of associated classes to be enumerated shall be determined using the following algorithm:

2150 • Initially, the set of classes to be enumerated is the set of all classes associated to any of
2151 the far ends of all associations referencing the starting class.

2152 • If the *AssociationClassName* operation input parameter is not NULL, it acts as a restricting
2153 filter on the classes to be enumerated such that each class that is associated with the
2154 starting class using an association class where the class or one of its superclasses does
2155 not have the name specified in *AssociationClassName*, is removed from the set of classes
2156 to be enumerated. There shall be no validity checking performed for the
2157 *AssociationClassName* operation input parameter; if the specified class does not exist, the
2158 operation shall succeed with an empty result (because the filter did not match).

2159 • If the *AssociatedClassName* operation input parameter is not NULL, it acts as a restricting
2160 filter on the classes to be enumerated such that each class where the class or one of its
2161 superclasses does not have the name specified in *AssociatedClassName*, is removed from
2162 the set of classes to be enumerated. There shall be no validity checking performed for the

AssociatedClassName operation input parameter; if the specified class does not exist, the operation shall succeed with an empty result (because the filter did not match).

NOTE Specifying a non-NULL value for *AssociatedClassName* ensures that the returned classes have the class specified in *AssociatedClassName* as a common superclass.

- If the *SourceRoleName* operation input parameter is not NULL, it acts as a restricting filter on the classes to be enumerated such that each class that is associated with the starting class using an association class that has a role name on its starting end that is not the role name specified in *SourceRoleName*, is removed from the set of classes to be enumerated. There shall be no validity checking performed for the *SourceRoleName* operation input parameter; if the specified role does not exist, the operation shall succeed with an empty result (because the filter did not match).
- If the *AssociatedRoleName* operation input parameter is not NULL, it acts as a restricting filter on the classes to be enumerated such that each class that is associated with the starting class using an association class that has a role name on the far end referencing that class that is not the role name specified in *AssociatedRoleName*, is removed from the set of classes to be enumerated. There shall be no validity checking performed for the *AssociatedRoleName* operation input parameter; if the specified role does not exist, the operation shall succeed with an empty result (because the filter did not match).

The consistency model defined in 5.8 applies.

The set of properties to be included in each returned associated class shall be determined using the following algorithm:

- The set of properties to be included is the set of properties exposed by the class. This includes all the duplicates of any duplicate non-overridden properties.

Preconditions:

- The namespace of the source class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- The namespace of any returned classes shall exist. If it does not exist, the operation shall fail, indicating WIPG0204. Note that cross-namespace association traversals may return classes in a server or namespace that is different from the server or namespace of the source class.

Postconditions:

- The associated classes with their class paths shall have been returned as described in the Description paragraph for this operation.
- Requirements on ACID properties:
  - Atomicity: N/A
  - Update Consistency: N/A
  - Isolation: Required at the level of single classes, as defined in 5.8.
  - Durability: N/A

2200 **Error messages:**

2201

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	For input namespace and namespace of returned class paths
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

2202

2203 ### 6.7.3 ReferenceClasses

2204 **Purpose:**

2205 Enumerate the association classes that reference a given source class and return their
2206 representations and class paths.

2207 **Operation Input Parameters:**

2208

Generic Name	Generic Type	Requirement	Description
ClassPath	ClassPath	Mandatory	Class path of the source class (Context Parameter)
AssociationClassName	ClassName	Mandatory	NULL, or name of the association class, acting as a restricting filter on the association classes
SourceRoleName	PropertyName	Mandatory	NULL, or name of the role on the starting end of the association, acting as a restricting filter on the association classes
IncludeQualifiers	boolean	Mandatory	Indicates whether qualifier values on any returned CIM elements are to be included, as defined in 6.2.1

2209

Operation Output Parameters:

Generic Name	Generic Type	Requirement	Description
ClassList	ClassSpecificationWithPath [ ]	Mandatory	Sequence of the CIM association classes

Description:

The *ReferenceClasses* operation traverses an association from a class on a starting end to classes on all of its far ends and returns the CIM association classes traversed together with their class paths.

The set of association classes to be enumerated shall be determined using the following algorithm:

- Initially, the set of classes to be enumerated is the set of all association classes referencing the starting class.
- If the *AssociationClassName* operation input parameter is not NULL, it acts as a restricting filter on the classes to be enumerated such that each association class where the class or one of its superclasses does not have the name specified in *AssociationClassName*, is removed from the set of classes to be enumerated. There shall be no validity checking performed for the *AssociationClassName* operation input parameter; if the specified class does not exist, the operation shall succeed with an empty result (because the filter did not match).

  NOTE Specifying a non-NULL value for *AssociationClassName* ensures that the returned classes have the class specified in *AssociationClassName* as a common superclass.
- If the *SourceRoleName* operation input parameter is not NULL, it acts as a restricting filter on the classes to be enumerated such that each association class that has a role name on its starting end that is not the role name specified in *SourceRoleName,* is removed from the set of classes to be enumerated. There shall be no validity checking performed for the *SourceRoleName* operation input parameter; if the specified role does not exist, the operation shall succeed with an empty result (because the filter did not match).

The consistency model defined in 5.8 applies.

The set of properties to be included in each returned association class shall be determined using the following algorithm:

- The set of properties to be included is the set of properties exposed by the association class. This includes all the duplicates of any duplicate non-overridden properties.

Preconditions:

- The namespace of the source class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- The namespace of any returned classes shall exist. If it does not exist, the operation shall fail, indicating WIPG0204. Note that cross-namespace association traversals may return classes in a server or namespace that is different from the server or namespace of the source class.

2246 **Postconditions:**

2247 • The association classes with their class paths shall have been returned as described in the
2248 Description paragraph for this operation.

2249 • Requirements on ACID properties:

2250 – Atomicity: N/A

2251 – Update Consistency: N/A

2252 – Isolation: Required at the level of single classes, as defined in 5.8.

2253 – Durability: N/A

2254 **Error messages:**

2255

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	For input namespace and namespace of returned class paths
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

2256

2257 6.8 Qualifier type operations

2258 This subclause defines server operations that deal with qualifier types. As defined in DSP0004, qualifier
2259 types represent the declarations of qualifiers, not their values.

2260 6.8.1 GetQualifierType

2261 **Purpose:**

2262 Retrieve a qualifier type.

2263 **Operation Input Parameters:**

2264

Generic Name	Generic Type	Requirement	Description
QualifierTypePath	QualifierTypePath	Mandatory	Qualifier type path of the qualifier type to be retrieved (Context Parameter)

2265

2266 **Operation Output Parameters:**

2267

Generic Name	Generic Type	Requirement	Description
QualifierType	QualifierType	Mandatory	Representation of the returned qualifier type

2268

2269 **Description:**

2270 The *GetQualifierType* operation retrieves a representation of the qualifier type referenced by
2271 *QualifierTypePath*.

2272 **Preconditions:**

2273 • The qualifier type referenced by *QualifierTypePath* shall exist. If it does not exist, the operation
2274 shall fail, indicating WIPG0215.

2275 • The namespace of the qualifier type referenced by *QualifierTypePath* shall exist. If it does not
2276 exist, the operation shall fail, indicating WIPG0204.

2277 **Postconditions:**

2278 • The representation of the qualifier type shall have been returned as described in the Description
2279 paragraph for this operation.

2280 • Requirements on ACID properties:

2281 – Atomicity: N/A

2282 – Update Consistency: N/A

2283 – Isolation: Required

2284 – Durability: N/A

2285 **Error messages:**

2286

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0215	Qualifier type not found	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

6.8.2 DeleteQualifierType

Purpose:

Delete a given qualifier type.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
QualifierTypePath	QualifierTypePath	Mandatory	Qualifier type path of the qualifier type to be deleted (Context Parameter)

Operation Output Parameters:

None.

Description:

The *DeleteQualifierType* operation deletes the qualifier type referenced by *QualifierTypePath*.

As defined in DSP0004, any namespace needs to contain qualifier types for the meta qualifiers and standard qualifiers, and may contain qualifier types for the optional qualifiers. Thus, deleting any required qualifier types from a namespace will render that namespace non-compliant to DSP0004.

Preconditions:

- The qualifier type referenced by *QualifierTypePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0215.
- The namespace of the qualifier type referenced by *QualifierTypePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- The qualifier identified by *QualifierTypePath* shall not be specified on any element in the same namespace. If this is not satisfied, the operation shall fail, indicating WIPG0233.

Postconditions:

- The qualifier type shall have been deleted as described in the Description paragraph for this operation.
- Requirements on ACID properties:
  - Atomicity: Required
  - Update Consistency: Required
  - Isolation: Required
  - Durability: Required

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0215	Qualifier type not found	Mandatory	Infrastructure	
WIPG0233	Qualifier type is used	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

6.8.3 ModifyQualifierType

Purpose:

Change a given qualifier type.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
QualifierTypePath	QualifierTypePath	Mandatory	Qualifier type path of the qualifier type to be changed (Context Parameter)
ModifiedQualifierType	QualifierType	Mandatory	Representation of the changed qualifier type

Operation Output Parameters:

None.

Description:

The *ModifyQualifierType* operation changes the qualifier type referenced by *QualifierTypePath*.

The qualifier type referenced by *QualifierTypePath* is replaced with the qualifier type representation specified in *ModifiedQualifierType*.

As defined in DSP0004, any namespace needs to contain qualifier types for the meta qualifiers and standard qualifiers, and may contain qualifier types for the optional qualifiers. Thus, changing these qualifier types in a namespace inconsistently with their DSP0004 definition will render that namespace non-compliant to DSP0004.

Preconditions:

- The qualifier type referenced by *QualifierTypePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0215.
- The namespace of the qualifier type referenced by *QualifierTypePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- The name of the qualifier type representation specified in *ModifiedQualifierType* shall equal the name of the qualifier type referenced by *QualifierTypePath*. If this is not satisfied, the operation shall fail, indicating WIPG0208.
- The request to modify the qualifier type shall satisfy any backward compatibility requirements defined in DSP0004. If this is not satisfied, the operation shall fail, indicating WIPG0234.
- If the qualifier type referenced by *QualifierTypePath* is one of the qualifiers defined in DSP0004, (i.e., meta, standard, and optional qualifiers), the new definition of the qualifier in *ModifiedQualifierType* shall be consistent with the definition of the qualifier in DSP0004. If this is not satisfied, the operation shall fail, indicating WIPG0245.

Postconditions:

- The qualifier type referenced by *QualifierTypePath* shall have been modified as defined in the Description paragraph for this operation.
- The backward compatibility requirements defined in DSP0004 shall be satisfied for the modified qualifier type.
- Requirements on ACID properties:
  - Atomicity: Required
  - Update Consistency: Required

– Isolation: Required

– Durability: Required

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0215	Qualifier type not found	Mandatory	Infrastructure	
WIPG0234	Incompatible modification of qualifier type	Mandatory	Infrastructure	
WIPG0245	Qualifier type inconsistent with DSP0004	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

6.8.4 CreateQualifierType

Purpose:

Create a qualifier type.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
NamespacePath	NamespacePath	Mandatory	Namespace path of the namespace in which the qualifier type is to be created (Context Parameter)
NewQualifierType	QualifierType	Mandatory	Representation of the qualifier type to be created

Operation Output Parameters:

Generic Name	Generic Type	Requirement	Description
QualifierTypePath	QualifierTypePath	Mandatory	Qualifier type path of the new qualifier type

Description:

The *CreateQualifierType* operation creates a qualifier type in the namespace referenced by *NamespacePath*, using the qualifier type representation specified in *NewQualifierType*, and returns the qualifier type path of the new qualifier type.

As defined in DSP0004, any namespace needs to contain qualifier types for the meta qualifiers and standard qualifiers, and may contain qualifier types for the optional qualifiers. Thus, creating these qualifier types in a namespace inconsistently with their DSP0004 definition will render that namespace non-compliant to DSP0004.

Preconditions:

- The namespace referenced by *NamespacePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- The qualifier type to be created shall not exist in the namespace referenced by *NamespacePath*. If this is not satisfied, the operation shall fail, indicating WIPG0248.
- If the qualifier type defined in *NewQualifierType* is one of the qualifiers defined in DSP0004, (i.e., meta, standard, and optional qualifiers), the definition of the qualifier in *NewQualifierType* shall be consistent with the definition of the qualifier in DSP0004. If this is not satisfied, the operation shall fail, indicating WIPG0245.

Postconditions:

- The qualifier type shall have been created as defined in the Description paragraph for this operation.
- Requirements on ACID properties:
  - Atomicity: Required
  - Update Consistency: Required
  - Isolation: Required
  - Durability: Required

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0248	Qualifier type already exists	Mandatory	Infrastructure	
WIPG0245	Qualifier type inconsistent with DSP0004	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

6.8.5 EnumerateQualifierTypes

Purpose:

Enumerate the qualifier types in a namespace.

Operation Input Parameters:

Generic Name	Generic Type	Requirement	Description
NamespacePath	NamespacePath	Mandatory	Namespace path of the namespace in which the qualifier types are to be enumerated (Context Parameter)

Operation Output Parameters:

Generic Name	Generic Type	Requirement	Description
QualifierTypeList	QualifierTypeWithPath [ ]	Mandatory	Sequence of the returned qualifier type representations and qualifier type paths

Description:

The *EnumerateQualifierTypes* operation enumerates all qualifier types in the namespace referenced by *NamespacePath*, and returns their representations and qualifier type paths.

Preconditions:

- The namespace referenced by *NamespacePath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.

Postconditions:

- The qualifier type representations and qualifier type paths shall have been returned as defined in the Description paragraph for this operation.
- Requirements on ACID properties:
  - Atomicity: N/A
  - Update Consistency: N/A
  - Isolation: Required at the level of single qualifier types, as defined in 5.8.
  - Durability: N/A

Error messages:

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

6.9 Indication delivery operations

This subclause defines listener operations that deal with the delivery of indications.

6.9.1 DeliverIndication

Purpose:

Deliver an indication to a listener.

2430 ## Operation Input Parameters:

2431

Generic Name	Generic Type	Requirement	Description
ListenerDestination	ListenerDestination	Mandatory	Address of the listener to which the indication will be delivered (see 5.4.20 for details) (Context Parameter)
Indication	InstanceSpecification	Mandatory	Representation of the indication instance

2432

2433 ## Operation Output Parameters:

2434 None

2435 ## Description:

2436 The *DeliverIndication* listener operation delivers the indication specified by *Indication* to the listener
2437 referenced by *ListenerDestination*.

2438 Reliable indication delivery as defined in DSP1054 is an optional part of the operation semantics.
2439 Generic operations mappings shall state whether reliable indication delivery is supported.

2440 ## Preconditions:

2441 • None

2442 ## Postconditions:

2443 • The indication shall have been delivered to the listener.

2444 • Requirements on ACID properties:

2445 – Atomicity: N/A

2446 – Update Consistency: N/A

2447 – Isolation: N/A

2448 – Durability: N/A

2449 ## Error messages:

2450

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0250	WBEM listener is shutting down	Optional	Infrastructure	
WIPG0251	WBEM listener limits are exceeded	Optional	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	This also covers timeout due to exhaustion of retries in reliable indication delivery (if supported).
WIPG0227	Other failure	Optional	Infrastructure	

2451

ANNEX A
(normative)

Cross-namespace associations

This annex describes cross-namespace associations, in order to define which instances and classes exist in which namespace in such a scenario, and what is to be returned by association traversal operations. This annex reflects the preconditions stated for the association traversal operations in this specification, but it defines additional rules for conforming implementations and is therefore normative.

In this annex, classes in a particular namespace are referred to using the syntax `<ns-name>::<class-name>` where `<ns-name>` is the namespace name and `<class-name>` is the class name. Instances in a particular namespace are referred to using the syntax `<ns-name>::<inst-name>` where `<ns-name>` is again the namespace name and `<inst-name>` is the name of the instance as stated in the diagram (which is purely a diagramming name and has nothing to do with its keys).

In this version of this document, this annex only covers the simple case of a binary association where both sides use the same schema version. More complex cases, e.g. of associations with more than two ends, or with schema different versions, are possible, but not covered in this version.

A.1 Binary association using same schema version

This subclause discusses a binary association (that is, an association with two ends) that crosses namespaces, and the two namespaces contain the same version of the schema.

Figure 4 is a UML class diagram showing the classes used by this scenario, in the typical drawing notation used in management profiles:

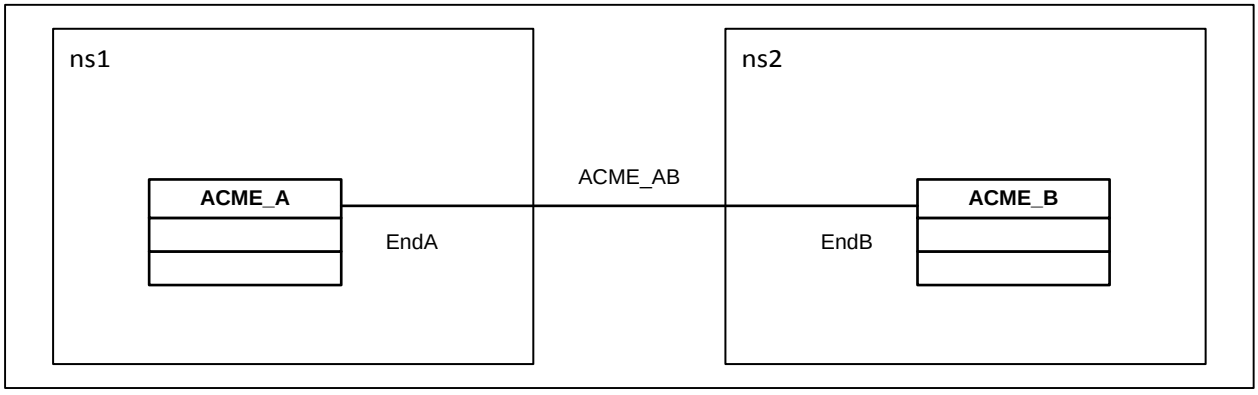

Figure 4 – Typical profile representation of binary association crossing namespaces

The namespaces (in this scenario, `ns1` and `ns2`) are shown as boxes around a number of classes used in the management profile. Class `ACME_A` is in namespace `ns1`, class `ACME_B` in namespace `ns2`. Association `ACME_AB` crosses between these two namespaces and is therefore termed a *cross-namespace association*.

This way of drawing the situation leaves it open exactly which classes and instances exist in each of the namespaces, and which of them are returned by instance-level and class-level association traversal operations.

Figure 5 is a UML structure diagram showing the classes and instances in a WBEM server that need to exist when the classes shown in Figure 4 are implemented for bidirectional association traversal.

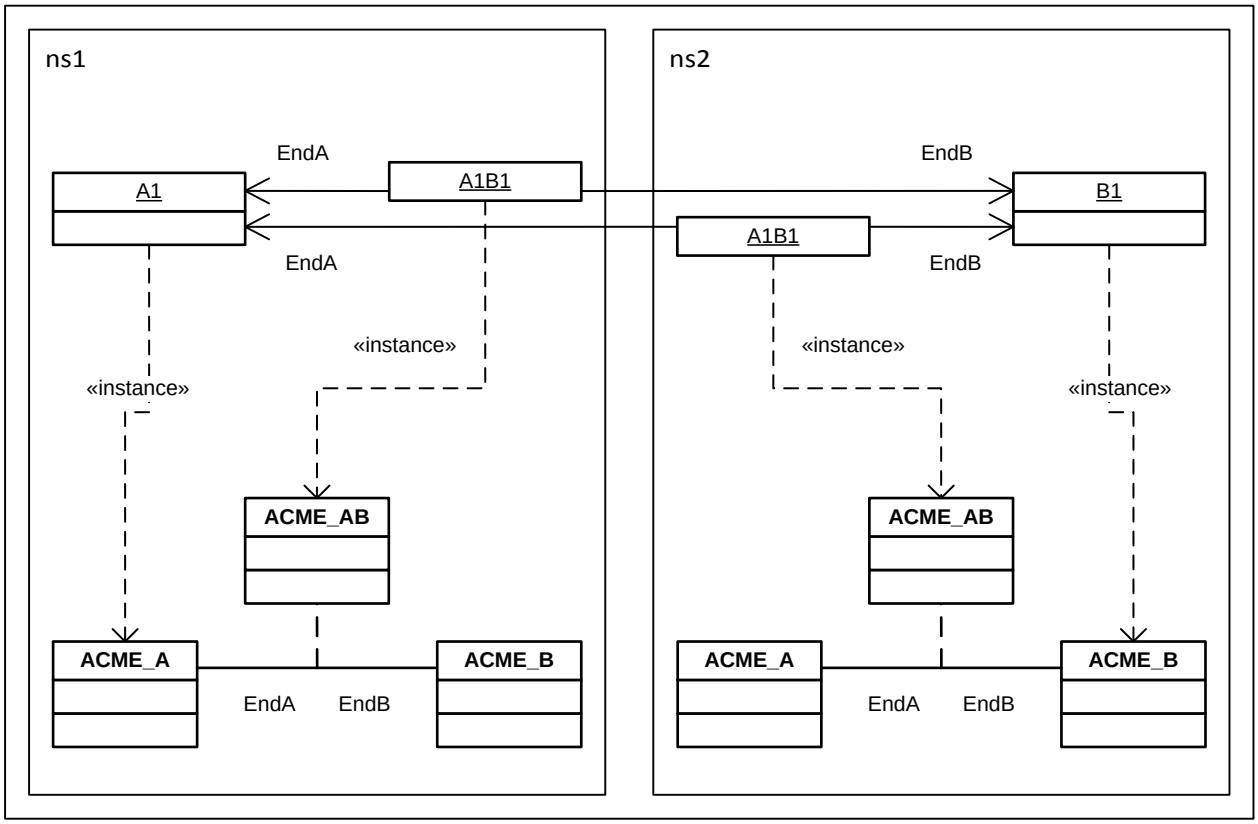

Figure 5 – Binary association: WBEM server objects for bidirectional traversal

The upper part of the figure shows instances, the lower part shows classes. Both of these are objects in a particular namespace of a WBEM server. Note that every object in this diagram is contained in a namespace. This is consistent with DSP0004 which defines that the name of every object (class, instance, qualifier type) has a namespace path component. As a result, the instance `A1B1` of the cross-namespace association `ACME_AB` appears in each of the two namespaces; in a way, it is duplicated.

Rule: Conformant implementations of bidirectional association traversal across namespaces shall have any such bidirectional cross-namespace association instances exist in both namespaces, and shall have the instances associated through such cross-namespace associations exist in only one namespace.

Enumerating the instances of the association class `ACME_AB` in namespace `ns1` (e.g. with the EnumerateInstances operation) returns instance `ns1::A1B1`, and enumerating the instances of `ACME_AB` in namespace `ns2` returns instance `ns2::A1B1`. This means that the association instances act like any other instances: They can be enumerated (if the operation is implemented) and that enumeration is scoped to a particular namespace. The instances `ns1::A1B1` and `ns2::A1B1` are distinct instances, because their namespace path is different.

The instances of the associated classes `ACME_A1` and `ACME_B1` appear only in their respective namespaces; they are not duplicated. As a result, enumerating the instances of `ACME_A` in namespace `ns1` returns `ns1::A1`, and enumerating the instances of `ACME_A` in `ns2` returns no instances (the EnumerateInstances operation still succeeds, because the class `ACME_A` exists in `ns2`).

The association traversal operations work in both directions in this scenario:

Traversing association `ACME_AB` starting from instance `ns1::A1` using the Associators operation results in instance `ns2::B1`, and traversing association `ACME_AB` starting from instance `ns2::B1` using the Associators operation results in instance `ns1::A1`. Because this behavior can be determined from the descriptions of these operations for the single-namespace case, no special rule for the cross-namespace case has been defined.

Because of the duplication of association instances in both namespaces, the situation is not intuitively clear for the References operation and other association-returning operations: The association instances `ns1::A1B1` and `ns2::A1B1` both reference the instance `ns1::A1`, so from a perspective of following the specified behavior for this operation by the letter, one can argue that both instances need to be returned, because they both exist and both reference the source instance. However, because these two instances are logically the same, a client would need to reduce the result set by eliminating such logical duplicates. Therefore, this annex defines the following restricting rule:

Rule: Conformant implementations of the References operation and other association-returning operations (such as ReferenceNames and OpenReferences) shall return only association instances that exist in the namespace of the source instance (even if a duplicated association instance exists in the other namespace).

For classes, the existence requirements are driven by their role as a creation class, and their role as a declared target of a reference in an association class. In Figure 5, the four classes that are the target of the `<<instance>>` dependency need to exist because they have instances in their namespace. Because the references in an association class only declare their targeted class but not their targeted namespace, schema consistency rules require that all classes referenced by an association class exist as objects in the same namespace as the association class. As a result, class `ACME_A` in addition needs to exist in `ns2`, and class `ACME_B` in addition needs to exist in `ns1`. Note that this is driven by consistency rules within a schema in a namespace, and is independent of whether or not class-level association traversal operations are supported. As a result, no additional rule needs to be defined for the existence of class objects in a cross-namespace case.

Because of the limitation that class-level references do not declare a target namespace, this annex defines the following rule for the behavior of the class-level operations:

Rule: Conformant implementations of the AssociatorClasses and ReferenceClasses operation shall return only classes that exist in the namespace of the source class; they never cross namespace boundaries.

Figure 6 shows the classes and instances in a WBEM server that need to exist when the classes shown in Figure 4 are implemented for unidirectional association traversal in the direction from `ns1` to `ns2`:

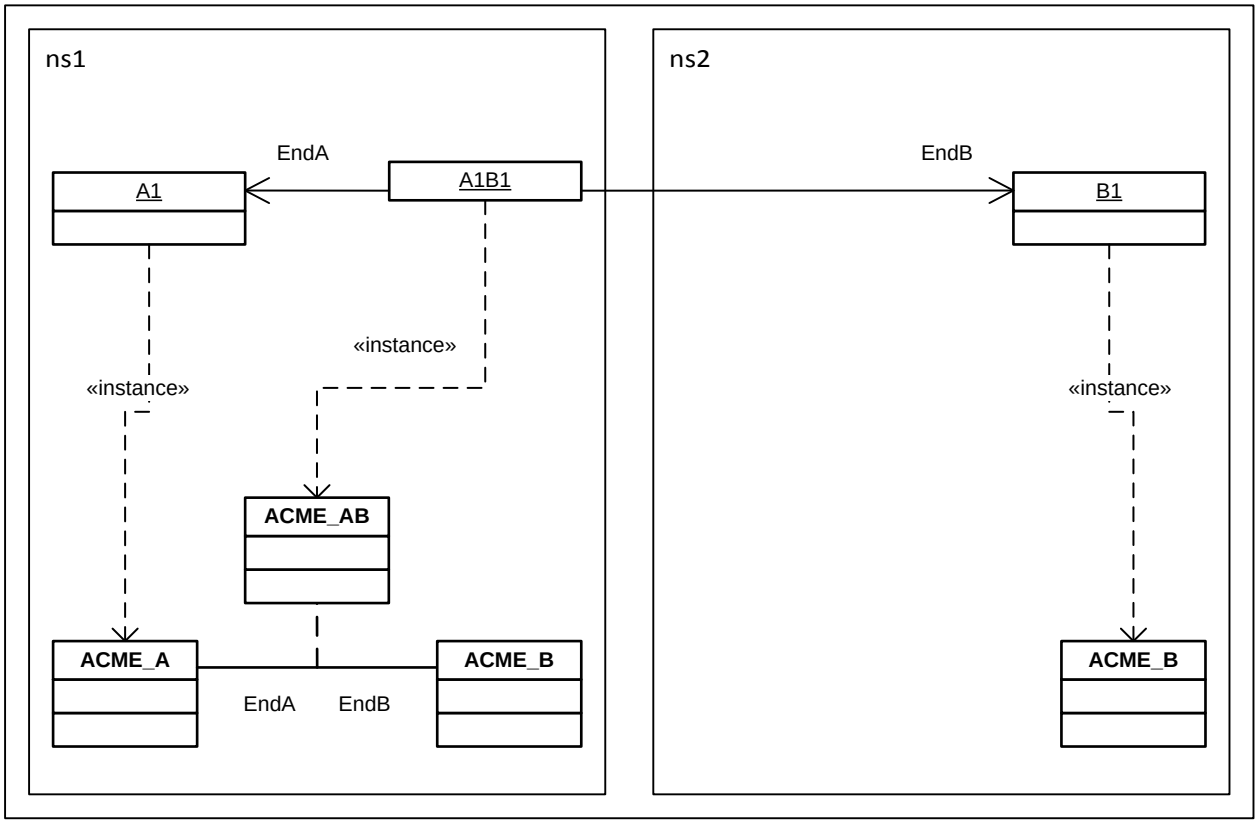

Figure 6 – Binary association: WBEM server objects for unidirectional traversal

In this case, the association instance `A1B1` only exists in namespace `ns1`, where traversal starts from.

Rule: Conformant implementations of unidirectional association traversal across namespaces shall have any such unidirectional cross-namespace association instances exist in only the source namespace where traversal starts from, and shall have the instances associated through such cross-namespace associations exist in only one namespace.

Because there is no instance `A1B1` in namespace `ns2`, there is no need for the classes `ACME_AB` and `ACME_A` to exist in `ns2`. As a result, namespace `ns2`, is "logically unaware" that namespace `ns1` can traverse into it. Whether this implies "implementation unawareness" depends on the type of WBEM server infrastructure that is used.

ANNEX B
(informative)

Change log

Version	Date	Description
1.0.0	2010-04-22	Published as DMTF Standard, with the following changes: • Consolidated terminology with DSP0004 2.6 and DSP1001 1.1. • Simplified the definition of generic types by relating them to DSP0004 2.6. • Clarifications for error handling and for pre- and postconditions. • Added definition of ACID properties and defined ACID requirements on all operations. • CreateInstance: Fixed incorrect statement about initial value if a property defines no default value in its class declaration. • ModifyClass: Removed message WIPG0232. • OpenQueryInstances: Removed message WIPG0124. • OpenAssociatedInstances…: Replaced message WIPG0214 with WIPG0213. • OpenReferencingInstances…: Replaced message WIPG0214 with WIPG0213. • GetAssociatedInstances..: Added message WIPG0213. • GetReferencingInstances..: Added message WIPG0213. • Removed ExecQuery operation and QueryResult type. • Removed GetAssociatedGraphInstancesWithPath and OpenAssociatedGraphInstancesWithPath and added operations for retrieval of associated instance graphs into ANNEX A (Future operations). • Stated the messages to be used for precondition violations. This affects all operations. • Added sources of messages (infrastructure / class implementation). This affects all operations. • Added usage of message WIPG0249 as needed and adjusted the name of message WIPG0208, to accommodate the DSP8016 change that splits message WIPG0208 into WIPG0208 and WIPG0249. This affects most operations. • Removed informative annex about required updates to other DMTF specifications. • Moved reference to DSP1001 into Bibliography • Changed terms: WBEM server, WBEM client, WBEM operation, WBEM protocol, WBEM listener, WBEM indication; Added references to document related terms in ISO guidelines. • Added "class implementation" as an additional source for error message WIPG0240 (WBEM server limits are exceeded) and WIPG0249 (Invalid input parameter value) on all instance related operations that use these messages • Generalized name of message WIPG0222 from "Query language feature not supported by WBEM server infrastructure" to "Query language feature not supported", following the corresponding change in DSP8016 1.0.1 • Clarified that error message source (class specific vs. infrastructure) is a recommendation only • Changed DeleteDependents parameter of DeleteClass operation to be experimental

Version	Date	Description
1.1.0	2015-01-07	Published as DMTF Standard, with the following changes: • Fixed an error in the description of the IncludeInheritedElements parameter of the GetSubClassesWithPath operation (it is based on the specified class, not on the returned classes). • Clarified why the GetTopClassesWithPath operation does not have an IncludeInheritedElements parameter. • Deprecated the IncludeProperties parameter of the GetClass, GetAssociatedClassesWithPath, and GetReferencingClassesWithPath operations, with no replacement. • In CreateInstance, fixed that property default values are now treated as an initialization constraint, together with the PropertyConstraint qualifier and constraints defined in management profiles. • In GetClass, fixed the Class output parameter to now be of type ClassSpecification (it was ClassSpecificationWithPath, returning the input class path again). • Added support for indications. • Added IncludeInheritedElements parameter to getClass. • Clarified that class origin information indicates the leaf-most class defining the element, in override situations, consistent with DSP0200 1.4. • Narrowed the definition of "WBEM protocol" in that it needs to conform to generic operations. • Improved the description of the interaction model and distinction between generic operations and WBEM protocol. • Updated the minor versions of several normative references, added DSP1054 as a normative reference, and moved DSP0228 to the Bibliography. • Wording improvements throughout the document. • From 1.0.2: Errata: Changed the names of the generic operations to be aligned with the CIM-XML operation names. See ANNEX B for details. • From 1.0.2: For the PullInstances operation, fixed an incorrect occurrence of its name, and an error in its description where it was incorrectly stated that it would return instances with path. • Added references to CIM-RS specifications. • Added requirement for WBEM protocols to support FQL (Filter Query Language). • Deprecated non-pulled instance operations that have pulled equivalents (EnumerateInstances, EnumerateInstanceNames, Associators, AssociatorNames, References, ReferenceNames). • Deprecated pulled instance operations returning instance paths (OpenEnumerateInstancePaths, OpenAssociatorPaths, OpenReferencePaths). • Deprecated EnumerationCount operation. • Deprecated IncludeClassOrigin input parameter on any instance operations (GetInstance, EnumerateInstances, Associators, References, OpenEnumerateInstances, OpenAssociators, OpenReferences). • Clarified which components have to be present in any object paths (namespace, instance, class, qualifier type). • Clarified requirements for existence of classes and namespaces of any class-level and instance-level (pulled and non-pulled) association operations, by adding according preconditions. • Errata: Fixed the missing WIPG0214 (Class not found) in pulled association operations. • Errata: Changed the behavior of (pulled and non-pulled) association operations in case the source instance does not exist, from failing to succeeding with an empty result set, in order to be aligned with the behavior of the corresponding CIM-XML operations. As a result, removed WIPG0213 (Instance not found) from their set of allowable error messages. • Errata: Removed the AssociatedClassName and AssociatedRoleName filters from (pulled and non-pulled) reference operations, in order to be aligned with the filtering abilities of the corresponding CIM-XML operations.

Version	Date	Description
		• Errata: Changed the name of parameter RoleName to SourceRoleName, of the class-level association operations, for consistency with the corresponding instance-level operations. • Added WIPG0240 (WBEM server limits are exceeded) to all operations that did not have it yet. • Added WIPG0214 (Class not found) to InvokeMethod operation. • Added ANNEX C, defining normative rules for cross-namespace associations. • Terms and abbreviations are now based on DSP0198.
2.0.0	2015-03-06	Published as DMTF Standard, with the following changes: • Removed the deprecated direct instance enumeration and association operations (EnumerateInstances, EnumerateInstanceNames, Associators, AssociatorNames, References, ReferenceNames). • Removed the deprecated pulled instance operations returning instance paths (OpenEnumerateInstancePaths, OpenAssociatorPaths, OpenReferencePaths and PullInstancePaths). • Removed the deprecated EnumerationCount operation. • Removed the class enumeration and association operations that return class names or class paths (EnumerateClassNames, AssociatorClassPaths, ReferenceClassPaths). • Removed the deprecated IncludeClassOrigin parameter from any instance operations. • Removed the ExcludeSubclassProperties parameter from any instance operations. • Removed the IncludeClassOrigin parameter from any class operations. • Removed the deprecated IncludedProperties parameter from any class operations. • Removed the annex about changed operation names. • Removed the annex about future ideas. • Removed alternative options for error handling, to require support for extended error handling using the generic error messages. • Clarified the meaning of requirement levels for generic error messages (see 5.7) and for operation parameters (see 5.3.3).

2557

Bibliography

DMTF DSP0200, *CIM Operations over HTTP 1.3*,
http://www.dmtf.org/standards/published_documents/DSP0200_1.3.pdf

DMTF DSP0201, *Representation of CIM in XML 2.3*,
http://www.dmtf.org/standards/published_documents/DSP0201_2.3.pdf

DMTF DSP0202, *CIM Query Language Specification 1.0*,
http://www.dmtf.org/standards/published_documents/DSP0202_1.0.pdf

DMTF DSP0203, *DTD for Representation of CIM in XML 2.3*,
http://www.dmtf.org/standards/published_documents/DSP0203_2.3.dtd

DMTF DSP0210, *CIM-RS Protocol 2.0*,
http://www.dmtf.org/standards/published_documents/DSP0210_2.0.pdf

DMTF DSP0211, *CIM-RS Payload Representation in JSON 2.0*,
http://www.dmtf.org/standards/published_documents/DSP0211_2.0.pdf

DMTF DSP0214, *Server Management Command Line Protocol Specification 1.0*,
http://www.dmtf.org/standards/published_documents/DSP0214_1.0.pdf

DMTF DSP0226, *Web Services for Management 1.0*,
http://www.dmtf.org/standards/published_documents/DSP0226_1.0.pdf

DMTF DSP0227, *WS-Management CIM Binding Specification 1.0*,
http://www.dmtf.org/standards/published_documents/DSP0227_1.0.pdf

DMTF DSP0228, *Message Registry XML Schema 1.1*,
http://schemas.dmtf.org/wbem/messageregistry/1/dsp0228_1.1.xsd

DMTF DSP0230, *WS-CIM Mapping Specification 1.0*,
http://www.dmtf.org/standards/published_documents/DSP0230_1.0.pdf

DMTF DSP1001, *Management Profile Specification Usage Guide 1.2*,
http://www.dmtf.org/standards/published_documents/DSP1001_1.2.pdf

DMTF DSP8028, *Management Profile XML Schema 1.1,*
http://schemas.dmtf.org/wbem/mgmtprofile/1/dsp8028_1.1.xsd

JCP JSR-48, *Java Community Process JSR-48: WBEM servers Specification*, not yet published,
http://jcp.org/en/jsr/detail?id=48

The Open Group CMPI, *Systems Management: Common Manageability Programming Interface (CMPI), Issue 2.0*, http://www.opengroup.org/bookstore/catalog/c061.htm