INTERNATIONAL STANDARD

ISO/IEC 19099

First edition
2014-05-01

Information technology — Virtualization Management Specification —

Technologies de l'information — Spécifications pour la gestion de la virtualisation

Reference number
ISO/IEC 19099:2014(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 19099 was prepared by SVPC Work Group of the DTMF (as INCITS 483-2012) and was adopted, under a special “fast-track procedure”, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by the national bodies of ISO and IEC.

American National Standard

for Information Technology –

Virtualization Management Specification

Developed by

incits SM

Where IT all begins

Approved American National Standard

ANSI

INCITS 483-2012

INCITS 483-2012

American National Standard
for Information Technology –

Virtualization Management Specification

Secretariat

Information Technology Industry Council

Approved May 29, 2012

American National Standards Institute, Inc.

American National Standard

Approval of an American National Standard requires review by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer.

Consensus is established when, in the judgement of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that a concerted effort be made towards their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give an interpretation of any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

CAUTION: The developers of this standard have requested that holders of patents that may be required for the implementation of the standard disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard. As of the date of publication of this standard and following calls for the identification of patents that may be required for the implementation of the standard, no such claims have been made. No further patent search is conducted by the developer or publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

Published by

American National Standards Institute, Inc.
25 West 43rd Street, New York, NY 10036

Printed in the United States of America

CONTENTS

Figures

Tables

Foreword (This foreword is not part of American National Standard INCITS 483-2012.)

Virtualization Management was prepared by the System Virtualization, Partitioning, and Clustering Working Group of the DTMF.

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.

Requests for interpretation, suggestions for improvement or addenda, or defect reports are welcome. They should be sent to InterNational Committee for Information Technology Standards (INCITS), ITI, 1101 K Street, NW, Suite 610, Washington, DC 20005.

This standard was processed and approved for submittal to ANSI by INCITS. Committee approval of this standard does not necessarily imply that all committee members voted for its approval. At the time it approved this standard, INCITS had the following members:

Don Wright, Chair
Jennifer Garner, Secretary

Organization Represented	*Name of Representative*
Adobe Systems Inc.	Scott Foshee
	Steve Zilles (Alt.)
AIM Global, Inc.	Steve Halliday
Apple	Helene Workman
	David Singer (Alt.)
Distributed Management Task Force	John Crandall
	Jeff Hilland (Alt.)
EMC Corporation	Gary Robinson
	Stephen Diamond (Alt.)
Farance, Inc.	Frank Farance
	Timothy Schoechle (Alt.)
GS1 US	Frank Sharkey
	Charles Biss (Alt.)
Hewlett-Packard Company	Karen Higginbottom
	Paul Jeran (Alt.)
IBM Corporation	Alexander Tarpinian
	Robert Weir (Alt.)
	Arnaud Le Hors (Alt.)
	Debra Boland (Alt.)
	Steve Holbrook (Alt.)
	Gerald Lane (Alt.)
IEEE	Jodie Haasz
	Terry deCourcelle (Alt.)
	Bob Labelle (Alt.)
Intel	Philip Wennblom
	Grace Wei (Alt.)
	Stephen Balogh (Alt.)
Lexmark International	Don Wright
	Dwight Lewis (Alt.)
	Paul Menard (Alt.)
	Jerry Thrasher (Alt.)
Microsoft Corporation	Jim Hughes
	Dick Brackney (Alt.)
	John Calhoon (Alt.)
National Institute of Standards & Technology	Michael Hogan
	Sal Francomacaro (Alt.)
	Dan Benigni (Alt.)
	Fernando Podio (Alt.)
	Teresa Schwarzhoff (Alt.)
	Wo Chang (Alt.)
	Elaine Newton (Alt.)

Organization Represented	*Name of Representative*
Oracle Corporation	Donald R. Deutsch
	Jim Melton (Alt.)
	Michael Kavanaugh (Alt.)
	Toshihiro Suzuki (Alt.)
	Jeff Mischkinsky (Alt.)
	Tony DiCenzo (Alt.)
	Patrick Curran (Alt.)
Purdue University	Stephen Elliott
	Kevin O'Connor (Alt.)
Telecommunications Industry Association (TIA)	Herb Congdon, II
	Cheryl Blum (Alt.)
US Department of Defense	Jerry Smith
	Dennis Devera (Alt.)
	Dave Brown (Alt.)
	Leonard Levine (Alt.)
	Matthew Young (Alt.)
	Thomas D'Agostino (Alt.)
US Department of Homeland Security	Peter Shebell
	Gregg Piermarini (Alt.)
	Juan Gonzalez (Alt.)
VMware	Lawrence Lamers
	Winston Bumpus (Alt.)

The System Virtualization, Partitioning, and Clustering Working Group of the DTMF, which developed this standard, had the following contributors:

Michael Johanssen, IBM, Editor

Contributors:

Gareth Bestor - IBM
Chris Brown - HP
Mike Dutch - Symantec
Jim Fehlig - Novell
Kevin Fox - Sun Microsystems, Inc.
Ron Goering - IBM
Daniel Hiltgen - EMC/VMware
Michael Johanssen - IBM
Larry Lamers - EMC/VMware
Andreas Maier - IBM
Aaron Merkin - IBM
John Parchem - Microsoft
Nihar Shah - Microsoft
David Simpson - IBM
Carl Waldspurger - EMC/VMware

Acknowledgments

The authors wish to acknowledge the following people.

Contributors:

- Kamesh Aiyer – EMC
- Tatyana Bagerman – Oracle
- Dave Barrett – Emulex
- Oliver Benke – IBM
- Gareth Bestor – IBM
- Chris Brown -- Hewlett-Packard
- Ron Doyle – IBM
- Mike Dutch – Symantec
- George Ericson – EMC
- James Fehlig – Novell
- Kevin Fox – Sun Microsystems, Inc.
- Michael Gering – IBM
- Ron Goering – IBM
- Steffen Grarup – VMware Inc.
- Steve Hand – Symantec
- Mark Hapner – Sun Microsystems, Inc.
- Daniel Hiltgen – VMware Inc.
- Michael Johanssen – IBM
- Mark Johnson – IBM
- Lawrence Lamers – VMware Inc.
- Richard Landau – Dell
- John Leung – Intel Corporation
- John Linn – EMC
- Fred Maciel – Hitachi Ltd.
- Andreas Maier – IBM
- Srinivas Maturi – Oracle
- Aaron Merkin – IBM
- John Parchem – Microsoft Corporation
- Shishir Pardikar – Citrix Systems Inc.
- Murali Rajagopal -- QLogic
- Rene Schmidt – VMware Inc.
- Stephen Schmidt – IBM
- Hemal Shah – Broadcom

- Nihar Shah – Microsoft Corporation
- David Simpson – IBM
- John Suit – Fortisphere
- Carl Waldsburger -- VMware Inc.
- Mike Walker – IBM
- Jeff Wheeler – Cisco

American National Standard
for Information Technology –

Virtualization Management Specification

Introduction

The information in this standard should be sufficient for a provider or consumer of this data to unambiguously identify the classes, properties, methods, and values that shall be instantiated to subscribe, advertise, produce, or consume an indication using the DMTF Common Information Model (CIM) Schema.

The target audience for this standard is implementers who are writing CIM-based providers or consumers of management interfaces that represent the components described in this document.

Document conventions

Typographical conventions

The following typographical conventions are used in this document:

- Document titles are marked in *italics*.
- Important terms that are used for the first time are marked in italics.
- ABNF rules are in `monospaced font`.

The following conventions are followed for defining formats of entries such as URIs:

- Literal characters within a format definition are surrounded by single quotes.
- Names of variables within a format are in standard text and are explicitly defined by means of a "Where: variable-name is …" section that follows the format definition.
- A specific value of a variable within a generalized example of a formatted entry is displayed in *italics*.
- Definitions of formats are case sensitive.
- Whitespace, if any, in formats is explicitly indicated.

In XML and MOF examples, an ellipsis ("`...`") indicates omitted or optional entries that would typically occupy the position of the ellipsis.

ABNF usage conventions

Format definitions in this document are specified using ABNF (see RFC 5234), with the following deviations:

- Literal strings are to be interpreted as case-sensitive Unicode characters, as opposed to the definition in RFC 5234 that interprets literal strings as case-insensitive US-ASCII characters.

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.

Experimental content is indicated by an Experimental Note.

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

DMTF component documents

Table 1 lists the DMTF component documents that were combined to create this standard.

Table 1 – Component documents

Document Number	Document Title	Version
DSP1041	*Resource Allocation Profile*	1.1.0
DSP1042	*System Virtualization Profile*	1.0.0
DSP1043	*Allocation Capabilities Profile*	1.0.0
DSP1044	*Processor Resource Virtualization Profile*	1.0.0
DSP1045	*Memory Resource Virtualization Profile*	1.0.0
DSP1047	*Storage Resource Virtualization Profile*	1.0.0
DSP1050	*Ethernet Port Resource Virtualization Profile*	1.0.0
DSP1057	*Virtual System Profile*	1.0.0
DSP1059	*Generic Device Resource Virtualization Profile*	1.0.0
DSP1097	*Virtual Ethernet Switch Profile*	1.0.0

1 Scope

1.1 Resource Allocation Profile

Clause 5 sets the basic resource allocation pattern for resource pools, allocations, and setting data. It also defines the resource-pool-lifecycle management and relationships.

1.2 System Virtualization Profile

Clause 6 is an autonomous profile that specifies the minimum top-level object model needed for the representation of host systems and the discovery of hosted virtual computer systems. In addition, it specifies a service for the manipulation of virtual computer systems and their resources, including operations for the creation, deletion, and modification of virtual computer systems and operations for the addition or removal of virtual resources to or from virtual computer systems.

1.3 Allocation Capabilities Profile

Clause 7 extends the management capability of referencing profiles by adding the ability to represent the default, supported and range of property values for resource allocation requests for a given resource, and the mutability of properties in a Resource Allocation Setting Data instance.

1.4 Processor Resource Virtualization Profile

Clause 8 is a component profile that extends the management capabilities of the specialized profiles by adding the support to represent and manage the allocation of processor resources to virtual systems.

1.5 Memory Resource Virtualization Profile

Clause 9 is a component DMTF management profile that extends the management capabilities of the referencing profile by adding the support to represent and manage the allocation of memory to virtual systems.

1.6 Storage Resource Virtualization Profile

Clause 10 is a component profile that extends the management capabilities of the referencing profile by adding the support to represent and manage the allocation of storage to virtual systems.

1.7 Ethernet Port Resource Virtualization Profile

Clause 11 is a component DMTF management profile that extends the management capabilities of the referencing profile by adding the support to represent and manage the allocation of Ethernet ports to virtual systems.

1.8 Virtual System Profile

Clause 12 is an autonomous profile that defines the minimum object model needed to provide for the inspection of a virtual system and its components. In addition, it defines optional basic control operations for activating, deactivating, pausing, or suspending a virtual system.

1.9 Generic Device Resource Virtualization Profile

Clause 13 is a concrete component profile that specializes the abstract Resource Allocation Profile described in clause 5 and the abstract Allocation Capabilities Profile described in clause 7.

Clause 13 is intended for use when a more specific resource allocation profile (for example, the Processor Resource Virtualization Profile described in clause 8, the Memory Resource Virtualization Profile described in clause 9, and so on) for common resource types has not yet been defined or approved, or when the device in question is an unusual device type for which no more specific profile exists.

1.10 Virtual Ethernet Switch Profile

Clause 14 is an autonomous DMTF management profile that defines the minimum object model needed to provide for the inspection of a virtualization system's internal Ethernet switch and its components.

2 Normative references

The following referenced documents are indispensable for the application of this document. 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, *CIM Schema 2.27*,
http://dmtf.org/standards/cim/schemas

DMTF DSP0004, *CIM Infrastructure Specification 2.6,*
http://www.dmtf.org/standards/published_documents/DSP0004_2.6.pdf

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

DMTF DSP0201, *Specification for the Representation of CIM in XML 2.3.1*,
http://www.dmtf.org/standards/published_documents/DSP201.pdf

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

DMTF DSP1001, *Management Profile Specification Usage Guide 1.0,*
http://www.dmtf.org/standards/published_documents/DSP1001_1.0.pdf

DMTF DSP1012, *Boot Control Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1012_1.0.pdf

DMTF DSP1014, *Ethernet Port Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1014_1.0.pdf

DMTF DSP1022, *CPU Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1022_1.0.pdf

DMTF DSP1026, *System Memory Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1026_1.0.pdf

DMTF DSP1027, *Power State Management Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1027_1.0.pdf

DMTF DSP1033, *Profile Registration Profile 1.0,*
http://www.dmtf.org/standards/published_documents/DSP1033_1.0.pdf

DMTF DSP1035, *Host LAN Network Port Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1035_1.0.pdf

DMTF DSP1052, *Computer System Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1052_1.0.pdf

DMTF DSP1053, *Base Metrics Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1053_1.0.pdf

DMTF DSP1054, *Indications Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1054_1.0.pdf

IETF RFC1738, *Uniform Resource Locators (URL)*, December 1994,
http://www.ietf.org/rfc/rfc1738.txt

IETF RFC3986, *Uniform Resource Identifier (URI): Generic Syntax*,
http://tools.ietf.org/html/rfc3986

IETF RFC5234, *ABNF: Augmented BNF for Syntax Specifications, January 2008*,
http://tools.ietf.org/html/rfc5234

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

SNIA SMI-S, *Storage Management Technical Specification* 1.3,
http://www.snia.org/tech_activities/standards/curr_standards/smi/SMI-S_Technical_Position_v1.3.0r5.zip

NOTE This standard refers to the following clauses of SNIA SMI-S: 1.3, Part 2 *Common Profiles*:

Clause 6: *Generic Target Ports* profile 1.0
Clause 14: *Generic Initiator Ports* profile 1.0

This standard refers to the following clauses of SNIA SMI-S: 1.3, Part 3 *Block Devices*:

Clause 5: *Block Services* package 1.3
Clause 15: *Extent Composition* subprofile 1.2

This standard refers to the following clauses of SNIA SMI-S: 1.3, Part 6 *Host Elements*:

Clause 6: *Storage HBA* profile 1.3
Clause 7: *Host Discovered Resources* profile 1.2

NOTE All parts of the SNIA SMI-S Storage Management Technical Specification have been approved as American National Standards, under the designation INCITS 388-2011. All parts of the specification are available at the ANSI Electronic Standards Store (ESS) on the ANSI website, www.ansi.org.

3 Terms and definitions

In this document, 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 document 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. Note that 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," and "annex" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 5.

The terms "normative" and "informative" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 3. In this document, clauses, subclauses, or annexes labeled "(informative)" do not contain normative content. Notes and examples are always informative elements.

The terms defined in DSP0004, DSP0200, and DSP1001 apply to this document. The following additional terms are used in this document.

3.1
allocated resource
the result of a resource allocation request — the assigned, separated, reserved, or shared part of the resource or emulated resource allocated to the consumer based on the resource allocation request

3.2
capability set
a set of instances of class CIM_SettingData associated with the association CIM_SettingsDefineCapabilities to a CIM_Capabilities instance, and the associated CIM_Capabilities instance.

3.3
child pool
pool whose resources are backed by other resource pools; consumer of resources from its parent resource pools; contains no host resources, instead draws resources from parent pools through resource allocations

3.4
client
an application that exploits facilities specified by the profiles in this standard

3.5
concrete memory resource pool
a resource pool that subdivides the capacity of its (primordial or concrete) parent resource pool

3.6
concrete storage resource pool
a storage resource pool that subdivides the capacity of its (primordial or concrete) parent resource pool

3.7
consumer
entity using allocated resources (for example, a virtual system)

3.8
current resource allocation setting data
resource allocation setting data that describes an allocated resource; differs from defined resource allocation setting data if the host system supports the dynamic modification of a resource allocation

3.9
current setting data
the virtual setting data associated with the current allocation state of a virtual resource or system.

3.10
dedicated virtual resource
virtual resource that has been given exclusive use of one or more host resources (the host resources are not shared with any other consumer)

3.11
defined resource allocation setting data
resource allocation setting data that describes a resource allocation request

3.12
dynamic ethernet connection allocation
an Ethernet connection in which a default Ethernet switch port is instantiated as part of an Ethernet connection allocation

3.13
ethernet adapter
an EthernetPort, its associated LAN Endpoint(s) and, optionally, a VLAN Endpoint that models the Ethernet device on a virtual or host system

3.14
ethernet adapter allocation request
a request for an Ethernet adapter resource allocation to a virtual machine; represented as instance of CIM_EthernetPortAllocationSettingData.

3.15
ethernet adapter resource allocation
the allocation of an Ethernet port to a virtual system

3.16
ethernet adapter resource pool
a resource pool that represents Ethernet adapters available as resources for a virtual computer system resource allocation

3.17
ethernet connection
the connection of two LAN endpoints where one LAN endpoint is implemented by an Ethernet adapter, and the other LAN endpoint is implemented by an Ethernet switch port, resulting in the connection of a virtual or host system Ethernet adapter to an Ethernet switch port

3.18
ethernet connection allocation request
an allocation request for a connection between a LAN Endpoint on an Ethernet adapter and a LAN Endpoint on an Ethernet switch port. An Ethernet connection allocation request may cause the implicit allocation of the entities that it connects, such as virtual Ethernet adapters and virtual switch ports. Ethernet connection allocation request is represented as instance of CIM_EthernetPortAllocationSettingData.

3.19
ethernet connection allocation
the allocation of an Ethernet connection between the LAN Endpoints of an Ethernet adapter and an Ethernet switch port

3.20
ethernet connection resource pool
a resource pool that represents available Ethernet connections on a virtual Ethernet switch for a virtual computer system

3.21
ethernet switch port
an EthernetPort, its associated LAN Endpoint(s) and, optionally, a VLAN Endpoint that models the Ethernet port on an Ethernet switch

3.22
ethernet switch port allocation request
a request for an Ethernet switch port resource allocation; represented as instance of CIM_EthernetPortAllocationSettingData.

3.23
ethernet switch port resource allocation
the allocation of an Ethernet port to a virtual Ethernet switch

3.24
ethernet switch port resource pool
a resource pool that represents Ethernet switch ports available as resources for a virtual Ethernet switch port resource allocation

3.25
host memory
a contiguous extent of memory contained by the host system that may be allocated with either exclusive or shared access to a memory resource pool

3.26
host processor resource
host processor resources are processor devices or computing resource contained by the host system that may be allocated with either exclusive or shared access to provide processing resources to a processor resource pool or a virtual system.

3.27
host resource
a device or computing resource contained by the host system that may be allocated with either exclusive or shared access through the host system to provide resources to a resource pool or consumer

3.28
host storage resource
a storage resource that exists in scope of or is accessible by a host system. A host system may contain or have access to one or more storage resources that may be as a whole or partially allocated to virtual systems

3.29
host system
the system that contains the host resources that are subject to resource allocation

3.30
implementation
a set of CIM providers that realize the classes specified by the profiles described in this standard

3.31
initiator port
a port that acts as the source for a data exchange operation

3.32
logical disk
the instantiation of allocated host resources that is exposed to a virtual system through a storage device; the result of a storage resource allocation based on a storage resource allocation request

3.33
memory composition
the aggregation of memory extents into an encompassing memory extent

3.34
memory resource allocation
the allocation of memory from a memory resource pool to a virtual system

3.35
memory resource allocation request
a request for a memory resource allocation

3.36
memory resource pool
a resource pool that represents memory available for memory resource allocation

3.37
memory resource pool configuration service
a configuration service that supports the addition or removal of host memory to or from a memory resource pool, and the creation or deletion of concrete subpools of a memory resource pool

3.38
port
communication endpoint for systems or storage devices. A port enables the exchange of data according to one or more protocols

3.39
primordial resource pool
a resource pool with no parent that may aggregate host resources

3.40
primordial storage resource pool
a storage resource pool that aggregates storage resources available for or used by storage resource allocations

3.41
processor resource
a processor device or computing resource as seen by a consumer

3.42
processor resource allocation
the allocation of a processor resource from a processor resource pool to a virtual system

3.43
processor resource allocation request
a request for a processor resource allocation

3.44
processor resource pool
a resource pool that represents processor resources available for processor resource allocation

3.45
processor resource pool configuration service
a configuration service that supports the addition or removal of host storage processor resources to or from a processor resource pool, and the creation or deletion of concrete subpools of a processor resource pool

3.46
resource allocation
process of assigning, separating, reserving, granting share of, or emulating resources for use by a consumer

3.47
resource allocation request
request for resources to be allocated

3.48
resource allocation setting data
RASD
CIM_ResourceAllocationSettingData - settings describing resource allocation, used by a host system to manage the allocation of resources and their relationship to host resources, resource pools used for the allocation, or both

3.49
resource pool
an abstract entity exposed by the virtualization platform for the purpose of allocation of allocated resources to consumers

3.50
resource type
generic type categorizing classes of resources (for example, processor, memory, network adapter)

3.51
shared virtual resource
virtual resource that has been given the use of host resources that may also be shared with other consumers

3.52
simple ethernet connection
an Ethernet connection in which a default Ethernet switch port and a default Ethernet adapter are instantiated as part of an Ethernet connection allocation

3.53
simple resource allocation
resource allocation with no logical device representing the allocated resources

3.54
static ethernet connection allocation
an Ethernet connection allocation where a specific pre-existing Ethernet switch port is requested as part of the allocation request

3.55
storage resource
a logical disk, a storage volume or a storage extent

3.56
storage resource allocation
the allocation of a storage resource from a storage resource pool to a virtual system

3.57
storage resource allocation request
a request for a storage resource allocation

3.58
storage resource pool
a resource pool that represents storage resources available for storage resource allocation

3.59
storage resource pool configuration service
a configuration service that supports the addition or removal of host storage resources to or from a storage resource pool, and the creation or deletion of concrete subpools of a storage resource pool

3.60
storage volume
the instantiation of allocated host resources that is exposed to a virtual system through a storage device that is published for use outside of the scoping system. Like a logical disk, a storage volume is the result of a storage resource allocation based on a storage resource allocation request

3.61
target port
a port that acts as a target of a data exchange operation

3.62
virtual computer system
virtual system as applied to a computer system
Other common industry terms for such a system include *virtual machine*, *hosted computer*, *child partition*, *logical partition*, *domain*, *guest*, and *container*.

3.63
virtual Ethernet switch
the concept of a virtual system as applied to a virtual Ethernet switch
A virtual Ethernet switch is a specialized virtual system.

3.64
virtual memory
the instantiation of allocated host memory that is exposed to a virtual system through a logical memory device; the result of a memory resource allocation based on a memory resource allocation request

NOTE The definition of the term “virtual memory” is specialized from the term “virtual resource” defined in clause 5 and deviates from common computer industry parlance.

3.65
virtualization platform
virtualizing infrastructure provided by a host system that enables the deployment of virtual systems

3.66
virtual processor
the instantiation of the allocated host processor resources that is exposed to a virtual system via a logical processor device.

3.67
virtual resource
an allocated resource that is represented as a logical device and assigned to a virtual system

3.68
virtual resource allocation
resource allocation with a logical device representing the allocated resources

3.69
virtual system
a computer system that is composed of virtual resources. Other common industry terms for such a system include *virtual machine*, *hosted computer*, *child partition*, *logical partition*, *domain*, *guest virtual machine*, and *container*.

3.70
virtualization platform
infrastructure that supports virtual systems. Other common industry terms for this infrastructure are *hypervisor* and *virtual machine monitor*.

4 Symbols and abbreviated terms

The abbreviations defined in DSP0004, DSP0200, and DSP1001 apply to this document. The following additional abbreviations are used in this document.

4.1
CIM
common information model

4.2
CIMOM
CIM object manager

4.3
CPU
central processing unit

4.4
EASD
ethernet port allocation setting data

4.5
ESD
element setting data

4.6
HBA
host bus adapter

4.7
MCA
capabilities settings of systems and of memory resource pools

4.8
MMS
mutability settings of memory resource allocation requests or memory resource allocations

4.9
MRA
memory resource allocation

4.10
MRQ
memory resource allocation request

4.11
RASD
resource allocation setting data

4.12
SASD
storage allocation setting data

4.13
SAN
storage area network

4.14
SDC
settings define capabilities

4.15
SDS
settings define state

4.16
SLP
service location protocol

4.17
SMI-S
Storage Management Initiative Specification

4.18
SNIA
Storage Networking Industry Association

4.19
VESSD
virtual ethernet switch setting data

4.20
VS
virtual system

4.21
VSSD
virtual system setting data

4.22
VSSDC
virtual system setting data component

5 Resource Allocation Profile

Profile Name: Resource Allocation

Version: 1.1.0

Organization: DMTF

CIM schema version: 2.22

Central Class: CIM_ResourcePool

Scoping Class: CIM_System

The Resource Allocation Profile is an abstract profile that extends the management capability of referencing profiles by adding the capability to represent the allocation of resources to consumers. This includes allocation of underlying supporting resources, such as power and cooling, and the allocation of computing resources, such as processors and memory. The resources may be virtualized. A general model is defined by the Resource Allocation Profile. Requirements and constraints specific to a resource type are defined in a referencing profile dedicated to the resource type. This profile defines a resource pool, allocated resources, allocation settings, and host resources.

The Resource Allocation Profile shall not be directly implemented. Implementation shall be based on a profile that specializes the requirements of this profile.

Table 2 identifies the profile on which the Resource Allocation Profile has a dependency.

Table 2 – Related profiles for the Resource Allocation Profile

Profile Name	Organization	Version	Requirement	Description
Allocation Capabilities	DMTF	1.0	Optional	Profile that describes allocation capabilities (see clause 7)

5.1 Description

This subclause provides an informative description of the management domain addressed by the profile described in this clause and describes how the CIM elements defined in the profile apply to the management domain.

Figure 1 is the class diagram for the Resource Allocation Profile. Cardinalities shown in the diagram reflect the constraints relative to implementations of this profile. For simplicity, the prefix *CIM_* has been removed from the names of the classes.

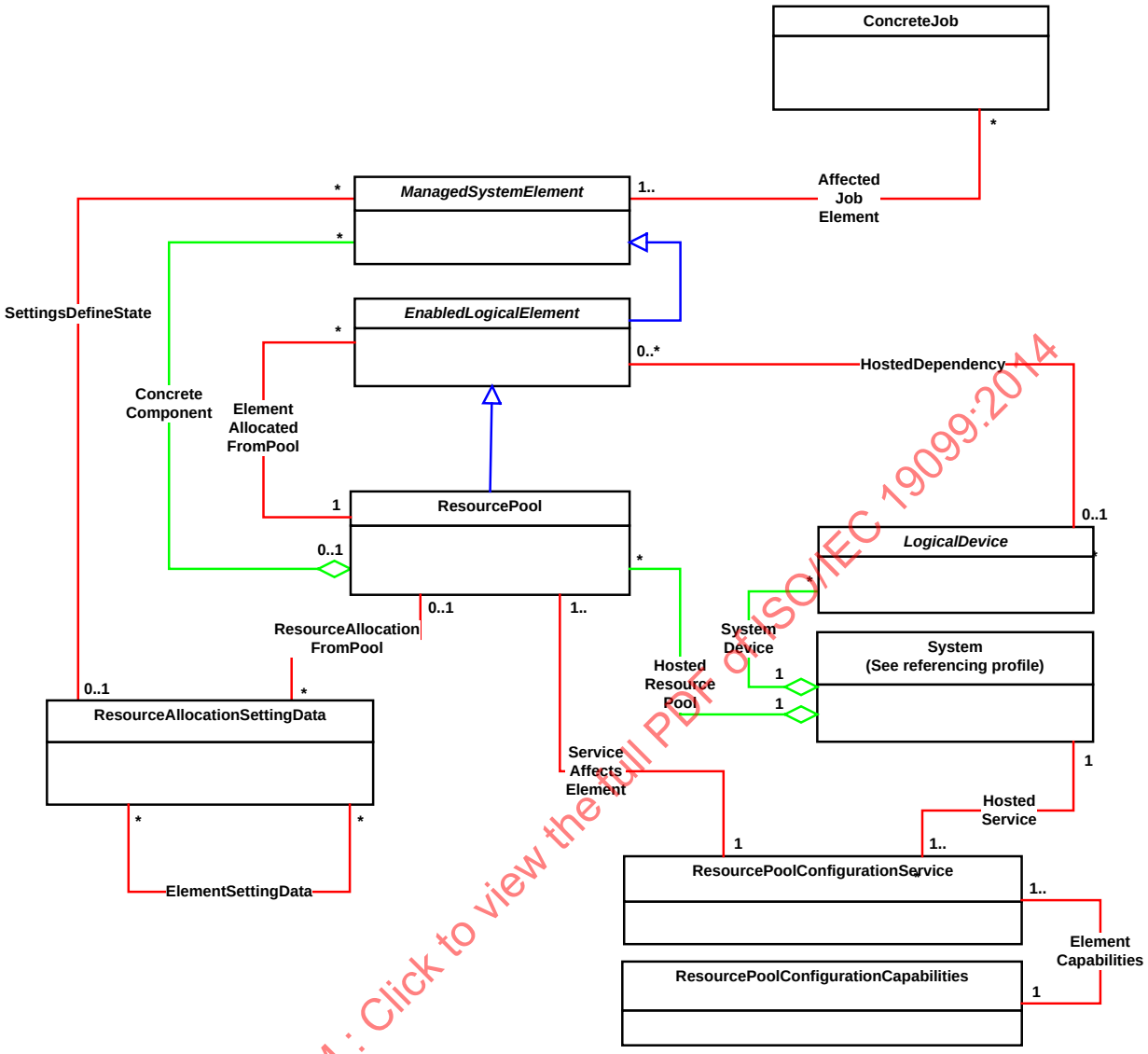

Figure 1 – Resource Allocation Profile: Class Diagram

5.1.1 General resource allocation concepts

The Resource Allocation Profile captures the general concept of defining the availability of a given resource type for allocation to consumers. The amount of resource available and the amount of resource allocated are modeled and managed. The aggregation of the underlying components that provide the resource (host resources) may be represented.

5.1.1.1 Host resources

Host resources are those that compose or enable a computer system. Examples include processors, memory, I/O, power supplied to the system, cooling allocated to the system, and so on.

5.1.1.2 Resource pool

The profile described in this clause uses a resource pool as the focal point for resource allocations. Consumers receive resource allocations from resource pools based on resource allocation requests. A resource pool results from aggregation of host resources of a specific type into a resource pool.

For example, when modeling virtualized computer systems, host processors are aggregated into a pool, giving it a known computing capacity, from which virtual processors are allocated. When modeling virtualized storage, the pool may map to a RAID volume or other storage-aggregating construct. Using power as an example, the host power supplies may be aggregated into a pool, thus establishing the total power available for allocation. Power is then allocated to dependent resource consumers from the pool.

5.1.1.3 Resource allocation

A resource allocation is a resource that is allocated from a resource pool. A resource allocation request is a request for a resource allocation. A resource allocation is obtained based on a corresponding resource allocation request. Both resource allocation and resource allocation request are represented through instances of the CIM_ResourceAllocationSettingData class.

5.1.1.4 Hierarchies of resource pools

A hierarchy of resource pools may be supported. A hierarchy may be used to provide administrative controls over the set of resources or to partition resources into disjoint sets.

For example, the aggregate processor capability of the computer system may be divided into child pools for individual departments or users.

A hierarchy of resource pools represents the same type of resource and is acyclic. Hierarchies of resource pools consisting of different types of resources are not defined by the profile described in this clause.

5.1.1.5 Pool and resource management

The creation, deletion, and management of resource pools and assignment of the host resources that they contain are covered by the profile described in this clause. The allocation of resources to a consumer is covered in derived profiles. The CIM_ResourcePoolConfigurationService class provides extrinsic methods for the management of resource pools.

5.1.2 Simple resource allocation

Simple resource allocation is the allocation of resources (for example, power, cooling, and so on) to a consumer where no logical device represents the resource allocated.

5.1.3 Virtual resource allocation

Virtual resource allocation is the allocation of resources (for example, processor, memory, and so on) to a consumer using an allocated resource. Virtual resource allocation extends the concept of resource allocation with the addition of semantics specific to virtualization. The virtual resources represent the consumer's view of the allocated resource, which enables management of the allocated resource in the context of management of the consumer. Additional functionality, such as the management of resource allocation definition or state, is introduced.

Multiple virtual resource allocations or virtual resource allocation requests may affect a single resource. For example, a virtual disk may be affected by the allocation of a storage extent and of bandwidth. A resource allocation or resource allocation request may affect one resource or a comprehensive set of resources. For example, a set of virtual processors may be modeled by one resource allocation request.

5.1.3.1 Virtual resource

A virtual resource represents the consumer's view of the allocated resource. In a processor example, the virtual processor represents the virtual resource and may be consumed by a virtual computer system. In a storage example, a storage volume represents the virtual resource and may be consumed by a physical or a virtual computer system.

The same CIM class is used to model both the host resource and the virtual resource. In many systems, virtualization-specific details are hidden from the consumer, and the consumer is presented with a virtual resource that looks no different than a host resource (that is, a resource that is not virtualized). The property values contained in the instance of a virtual resource reflect what the consumer of that resource is presented, not the virtualization-specific details of the allocation.

Information specific to resource allocation is modeled using resource allocations and resource allocation requests. These are represented by instances of the CIM_ResourceAllocationSettingData class. This approach ensures that general-purpose management applications may consume information about the virtual resources without having to know details of the underlying virtualization. This approach also prevents proliferation of virtualization-specific properties throughout the CIM schema, or mandatory subclassing of every possible device that may be virtualized. For example, the virtualized processor in the virtual computer system uses the same properties as a physical processor in a physical computer system.

Resource allocation CIM management profiles that specialize the profile described in this clause may allow or require different CIM classes to be used to represent the virtual and host resources. For example, a consumable storage device may be modeled using CIM_StorageVolume or CIM_LogicalDisk, while the hosting resource may be modeled using CIM_StorageExtent.

5.1.3.2 Dedicated virtual resources

Dedicated virtual resources are allocated to a consumer and are not available to other consumers. The host resource backs the virtual resource through a one-to-one mapping that is identified by the MappingBehavior property of CIM_ ResourceAllocationSettingData set to a value of 2 (Dedicated).

5.1.3.3 Shared virtual resources

Shared resources may be used by multiple consumers. The host resource may map to multiple virtual resources that may be allocated to the same resource consumer. One host resource is shared by many consumers (for example, a quorum drive in a cluster environment).

A virtual resource may map to different host resources over time. One virtual resource may be mapped to many backing host resources (for example, a virtual processor scheduled to run on different host processors during the course of execution).

5.1.3.4 Relationship between host resource and virtual resource

If the virtual resource always maps to the same host resource, the CIM_HostedDependency association may be used to reflect this relationship for a current allocation. Implementations that support scheduling across the pool of host resources transparent to the consumer may not expose the CIM_HostedDependency association because this relationship may change frequently.

5.1.3.5 Resource allocation definition and resource allocation state

For each resource that may be allocated, a resource allocation request represents the resource allocation definition. Once one or more elements requested by the resource allocation definition are allocated, a corresponding set of resource allocations represents the resource allocation state. Details of a resource allocation request or a resource allocation are represented by an instance of the CIM_ResourceAllocationSettingData class. Elements of resource allocation definition and resource allocation state may be changed independently.

For example:

- A system may support changing the processor resource allocation for the next boot or reset of the virtual system while the system is running without changing the current state of the system. This behavior is reflected by updates to the defined settings.
- An implementation for virtual memory may support changing the resource allocation definition only. A change such as an increase in virtual memory becomes effective the next time the virtual system is activated.
- An implementation may support changing the settings for the currently running virtual system only where the values revert to the prior settings on the next boot or reset. This behavior is reflected by updates in the current settings only.
- An implementation may support changing the resource allocation definition and state of virtual processors simultaneously. A change such as an increase in the relative share of processing power that a group of processors scoped by a virtual system receive becomes effective immediately without a need for a re-activation, and remains effective beyond the next re-activation.

The kind of changes that are supported are defined in resource allocation CIM management profiles that specialize the profile described in this clause.

5.2 Implementation

This subclause provides normative requirements related to the arrangement of instances and properties of instances for implementations of the profile described in this clause. The CIM Schema definitions and requirements apply.

5.2.1 Common requirements

This subclause details requirements that shall be met regardless of whether simple resource allocation or virtual resource allocation is implemented. In addition to these common requirements, either the requirements in 5.2.2 or the requirements in 5.2.3 shall also be implemented.

5.2.1.1 Representation of a resource pool

Each resource pool managed using the profile described in this clause shall be represented by an instance of the CIM_ResourcePool class. It shall be associated with the instance of CIM_System that represents the scoping system through one instance of the CIM_HostedResourcePool association. Each resource pool shall represent resources of the same type.

5.2.1.2 Primordial and concrete resource pools

A resource pool may be primordial or concrete.

Primordial pools aggregate capacity; they represent the known manageable capacity for the host system. Capacity is drawn from the primordial pool to create concrete resource pools or to allocate resources to consumers. There shall be at least one primordial pool for each resource type managed through the profile described in this clause. The instance of the CIM_ResourcePool class that represents a primordial resource pool shall have the Primordial property set to a value of TRUE.

Concrete resource pools subdivide the resource capacity available at a system. A single concrete pool may represent all the capacity of a primordial pool. The instance of the CIM_ResourcePool class that represents a concrete resource pool shall have the Primordial property set to a value of FALSE.

If a one-to-one correspondence exists between the host resource and the virtual resource, the CIM_HostedDependency association may be used to indicate the correspondence.

5.2.2 Modeling virtual resource allocation

Virtual resource allocation may be modeled. If virtual resource allocation is modeled, the requirements specified in this subclause shall be met.

5.2.2.1 Host resources

Host resources are modeled as a subclass of the CIM_LogicalDevice class. Host resources may be aggregated into one or more primordial resource pools and allocated to resource pools or resource consumers.

If aggregation of host resources is supported, at least one instance of the CIM_LogicalDevice class shall be associated with the instance of the CIM_ResourcePool class through an instance of the CIM_Component association. If aggregation of host resources is supported, the CIM_ResourceAllocationSettingData.Capacity property shall be supported.

If a resource pool is used for dedicated or shared resources, aggregation of host resources should be supported.

5.2.2.2 Virtual resources

Each virtual resource that is fully or partially allocated shall be represented by an instance of the CIM_LogicalDevice class. That instance shall be associated with the instance of the CIM_System class that represents the scoping virtual system through an instance of the CIM_SystemDevice association.

5.2.2.3 Resource allocation definition

Each resource shall have a resource allocation definition. Each element of a resource allocation definition shall be represented by one instance of the CIM_ResourceAllocationSettingData class.

5.2.2.4 Resource allocations

An instance of the CIM_LogicalDevice class that represents a virtual resource shall be associated to zero or more instances of the CIM_ResourceAllocationSettingData class that represents the resource allocation state through an instance of the CIM_SettingsDefineState association.

One instance of the CIM_ResourceAllocationSettingData class may be associated with more than one instance of the CIM_LogicalDevice class that represents a set of virtual resources.

An instance of the CIM_ResourceAllocationSettingData class that represents the current allocation state shall be associated with a virtual resource through an instance of the CIM_SettingsDefineState association.

An instance of the CIM_ResourceAllocationSettingData class that represents the defined allocation state shall be associated with the instance of the CIM_ResourceAllocationSettingData class that represents the current allocation state through an instance of the CIM_ElementSettingData association with the IsDefault property set to 1 (Is Default).

The non-key properties of the two instances of the CIM_ResourceAllocationSettingData class may match if both the current and defined settings are the same.

An instance of the CIM_ResourceAllocationSettingData class that represents a current resource allocation shall be associated with one instance of the CIM_ResourcePool class through an instance of the CIM_ResourceAllocationFromPool association.

An instance of CIM_ResourceAllocationSettingData that represents a defined resource allocation shall not be associated with instances of CIM_ResourcePool through the CIM_ResourceAllocationFromPool association.

5.2.2.5 Dedicated allocations

If the value of the MappingBehavior property is set to 2 (Dedicated) in the instance of the CIM_ResourceAllocationSettingData class that represents the defined allocation state, and if no values are specified for the HostResource[ ] array property or the HostResource[ ] array property is not specified (NULL), the system shall select the host resources if the virtual resource is allocated.

If the value of the MappingBehavior property is set to 2 (Dedicated) in the instance of the CIM_Resource-AllocationSettingData class that represents the current allocation state, the HostResource[ ] array property shall contain the identities of host resources that are dedicated to the virtual resource. For dedicated resources, an instance of the CIM_HostedDependency association may be present between the instance of the CIM_LogicalDevice class that represents a dedicated host resource allocation and the instance of the CIM_LogicalDevice class that represents the virtual resource.

5.2.2.6 Allocations with affinity

Virtual resources may be allocated with affinity to host resources using values of the MappingBehavior property.

If the MappingBehavior property is set to 4 (Hard Affinity), only the resources specified in the HostResource[ ] array property shall be used. If no values are specified for the HostResource[ ] array property or the HostResource[ ] array property is not specified (NULL), the system shall select the host resources if the virtual resource is allocated and maintain the allocation of those resources to the virtual device.

If the MappingBehavior property is set to 3 (Soft Affinity), the resources specified in the HostResource[ ] array property are preferred, but alternative host resources may be used. If no values are specified for the HostResource[ ] array property or the HostResource[ ] array property is not specified (NULL), the system shall select the host resources if the virtual resource is allocated.

If the MappingBehavior property is set to 3 (Soft Affinity) or 4 (Hard Affinity), an instance of the CIM_HostedDependency association shall not be used between the instance of the CIM_LogicalDevice class that represents a dedicated host resource allocation and the instance of the CIM_LogicalDevice class that represents the virtual resource.

If values are specified for the HostResource[ ] array property, the number of resources listed in the HostResource[ ] array property shall be adequate to satisfy the allocation request but may include additional resources.

The HostResource[ ] array property that represents the defined allocation state shall be set to the user's request. The HostResource[ ] array property that represents the current allocation state shall be set to the current active behavior.

5.2.3 Modeling simple resource allocation

Simple resource allocation may be modeled. If simple resource allocation is modeled, the requirements in this subclause shall be met.

5.2.3.1 General requirements

Each instance of the CIM_ResourceAllocationSettingData class that represents a current allocation state or alternate allocation state shall be associated with one instance of the CIM_ResourcePool class through an instance of the CIM_ResourceAllocationFromPool association.

A logical device shall not be instantiated.

5.2.3.2 Current allocation

An instance of the CIM_ResourceAllocationSettingData class that represents the current state shall be associated with one instance of the CIM_ResourceAllocationSettingData class through an instance of the CIM_ElementSettingData association with the IsCurrent property set to 1 (Is Current). If the CIM_ResourceAllocationSettingData class that represents the current state is modified, the IsCurrent property shall be set to a value other than 1 (Is Current).

5.2.3.3 Alternate allocations

Alternate allocations of the resource for the consumer may be supported. Each alternate allocation state shall be represented by an instance of the CIM_ResourceAllocationSettingData class that is associated with an instance of the CIM_ResourceAllocationSettingData class that represents the alternate allocation state through an instance of the CIM_ElementSettingData association with the IsCurrent property set to 2 (Is Not Current).

5.2.4 Resource pool management

Resource pool management may be modeled. If resource pool management is modeled, the requirements of this subclause shall be met. Implementations may support active management of instances of the CIM_ResourcePool class, or they may expose a read-only view of existing instances of the CIM_ResourcePool class.

An instance of the CIM_ResourcePoolConfigurationService class shall be implemented; however, the methods of the service are optional. The instance of the CIM_ResourcePoolConfigurationService class shall be associated with the host system through an instance of the CIM_HostedService association. One instance of CIM_ResourcePoolConfigurationCapabilities shall be associated with the CIM_ResourcePoolConfigurationService instance through the CIM_ElementCapabilities association. This instance of CIM_ResourcePoolConfigurationCapabilities shall reflect the methods supported. If active management is not supported by an implementation, all properties of the associated CIM_ResourcePoolConfigurationCapabilities instance shall be set to NULL.

5.2.5 Metrics

If metrics are implemented, the DSP1053 shall be implemented. If the instance of the CIM_BaseMetricDefinition class defines a metric that applies across the entire resource pool, the instance of CIM_BaseMetricDefinition class shall be associated with an instance of the CIM_ResourcePool class through the CIM_MetricDefForME association, and the instance of the CIM_BaseMetricDefinition class shall not be associated with any other instances of the CIM_ManagedElement class. An example of this type of metric is a metric that reports the total instantaneous resource consumption from the pool.

If the instance of the CIM_BaseMetricDefinition class defines a metric related to an individual virtual device's utilization of resources from the resource pool, the instance of the CIM_BaseMetricDefinition class shall be associated with the instance of the CIM_ResourcePool class through the CIM_MetricDefForME association, and the instance of the CIM_BaseMetricDefinition class shall be associated with the instance of the CIM_LogicalDevice class that represents the virtual device through an instance of the CIM_MetricDefForME association.

If the instance of the CIM_BaseMetricDefinition class defines a metric for the virtual device that is not related to the consumption by the device of resources from the resource pool, the instance of the CIM_BaseMetricDefinition class shall not be associated with the instance of the CIM_ResourcePool class.

If the instance of the CIM_BaseMetricDefinition class defines a metric related to the resource pool and a host resource, the instance of the CIM_BaseMetricDefinition class shall be associated with the instance of the CIM_ResourcePool class through an instance of the CIM_MetricDefForME association, and the

instance of the CIM_BaseMetricDefinition class shall be associated with the instance of the CIM_ManagedElement class that represents the host resource through an instance of the CIM_MetricDefForME association.

5.2.6 Resource pool hierarchies

Hierarchies of resource pools may be modeled. A hierarchy of resource pools represents the same type of resource and shall be acyclic.

Child pools may be allocated from the parent pool using the CIM_ResourcePoolConfigurationService class.

Parent and child pools shall be scoped to the same system.

A pool may have virtual resources and child pools allocated from it simultaneously.

5.2.7 Virtual resource definition and modification

The CIM_ResourceAllocationSettingData class is used as an input for virtual system definition. The client and implementation considerations are defined. The Resource Allocation Profile specifies how to define and modify virtual resources using methods of the virtual system management service. In these method specifications, the CIM_ResourceAllocationSettingData class is used for parameterization of resource-allocation-specific properties. The capabilities model may be used to convey information about limitations for and default values of properties of the CIM_ResourceAllocationSettingData class; see 7.4.

5.2.7.1 CIM_ResourceAllocationSettingData.InstanceID property

A client shall set the InstanceID property to NULL if the instance of the CIM_ResourceAllocationSettingData class is created locally. A client shall not modify the InstanceID property in an instance of the CIM_ResourceAllocationSettingData class that was received from an implementation and is sent back to the implementation as a parameter of a modification method.

An implementation shall ignore any non-NULL value in a definition request.

In a modification request, an implementation shall use a non-NULL value to identify an existing instance of the CIM_ResourceAllocationSettingData class. If a value is specified that does not identify an instance of the CIM_ResourceAllocationSettingData class, an implementation shall return a return code that indicates an invalid parameter; see 5.3.

5.2.7.2 CIM_ResourceAllocationSettingData.ResourceType property

A client shall set the value of the ResourceType property to designate the type of the virtual resource allocation request.

The implementation shall use the value of the ResourceType property, as well as the value of the OtherResourceType property if the value of the ResourceType property is 1 (Other), to determine the type of the virtual resource allocation request. If the implementation does not support the requested resource type, it shall fail the method execution.

5.2.7.3 CIM_ResourceAllocationSettingData.OtherResourceType property (conditional)

If a client sets the value of the ResourceType property to a value other than 1 (Other), it shall set the value of OtherResourceType property to NULL. If a client sets the value of the ResourceType property to 1 (Other), it shall set the value of the OtherResourceType property to identify the type of the virtual resource allocation request in an implementation-dependent way.

The implementation shall use the value of the OtherResourceType property if the value of the ResourceType property is 1 (Other) to determine the type of the virtual resource allocation request. If the implementation does not support the requested resource type, it shall fail the method execution.

5.2.7.4 CIM_ResourceAllocationSettingData.ResourceSubType property

A client may set the value of the ResourceSubType property to designate the subtype of the virtual resource allocation request. A client may set the value of the ResourceSubType property to NULL, requesting default behavior.

The implementation shall use the value of the ResourceSubType property to determine the subtype of the virtual resource allocation request. If the implementation does not support the requested virtual resource subtype, it shall return a return code that indicates an invalid parameter; see 5.3.

5.2.7.5 CIM_ResourceAllocationSettingData.PoolID property

A client may set the value of the PoolID property to designate a resource pool that shall be used for resource allocation. In this case, the values of the PoolID and ResourceType properties shall be equal to the values of respective properties in an instance of the CIM_ResourcePool class that represents the designated resource pool. A client may set the value of the PoolID property to NULL, requesting default behavior.

An implementation shall use the value of the PoolID and ResourceType properties to assign the resource pool that shall be used for the resource allocation. If the value of the PoolID property is NULL, the implementation may assign a default resource pool. If no resource pool exists with matching values of the PoolID and ResourceType properties, the implementation may either assign a default resource pool or fail the method execution. An implementation may defer the selection of a default resource pool until resource allocation actually occurs.

5.2.7.6 CIM_ResourceAllocationSettingData.ConsumerVisibility property

A client may set the value of the ConsumerVisibility property to specify whether the virtual resource or comprehensive set of virtual resources that is requested by the virtual resource allocation request shall be virtualized or shall be one or more passed-through host resources. A client may specify a value of NULL for the ConsumerVisibility property, requesting a default behavior.

If the property is set to a value other than NULL, the client shall perform one of the following actions:

- set a value of 0 (Unknown) to request default behavior (with the same effect as a value of NULL)
- set a value of 2 (Passed-Through) to specify that one or more passed-through host devices shall be allocated to the virtual resource requested by this virtual resource allocation request, and shall provide one or more elements in the HostResource[ ] array property that identify the host resources that shall be passed through
- set a value of 3 (Virtualized) to specify that the virtual resource that results from this virtual resource allocation request shall be virtualized

The client shall not use a value of 4 (Not Represented).

An implementation shall use the value of the ConsumerVisibility property to determine whether the virtual resource or comprehensive set of virtual resources requested by this virtual resource allocation request shall be virtualized or shall be a passed-through host resource.

- If the value of the ConsumerVisibility property is NULL or 0 (Unknown), the implementation may exhibit an implementation-specific default behavior that may also depend on the resource type, the selected resource pool, or both.

- If the value of the ConsumerVisibility property is 2 (Passed-Through), the implementation shall establish a virtual resource allocation request or virtual resource allocation for host resources as specified by elements of the HostResource[ ] array property as passed-through devices in the resulting virtual resource allocation. If no values are specified by the HostResource[ ] array property, the implementation may exhibit an implementation-specific default behavior.
- If the value of the ConsumerVisibility property is 3 (Virtualized), the implementation shall establish a virtual resource allocation request or virtual resource allocation for a virtualized virtual device or a comprehensive set of virtualized virtual devices.
- If the value of the ConsumerVisibility property is 4 (Not Represented), the implementation shall fail the method execution.

5.2.7.7 CIM_ResourceAllocationSettingData.HostResource[ ] Array property

A client may set the value of the HostResource[ ] array property to indicate that the requested virtual resource allocation shall be based on host resources that are identified by element values. The kind of dependency is specified through the ConsumerVisibility property (see 5.2.7.6) and the MappingBehavior property (see 5.2.7.18). A client may set the value of the HostResource[ ] array property to NULL or may specify an empty array in order to request the implementation to decide whether the requested resource allocation shall be directly based on host resources.

An implementation shall use the value of the HostResource[ ] array property to determine whether and how the requested virtual resource allocation shall be based on host resources. Respective host resources are identified by element values of the HostResource[ ] array property. The implementation shall use the value of other properties such as ConsumerVisibility (see 5.2.7.6) and MappingBehavior (see 5.2.7.18) to determine the kind of dependency. If no value or an empty array is provided as the value of the HostResource[ ] array property, the implementation may provide a pure virtual resource for the request or may select host resources at its own discretion.

5.2.7.8 CIM_ResourceAllocationSettingData.AllocationUnits property

A client should set the value of the AllocationUnits property to specify a unit of measurement for the virtual resource allocation request. The unit of measurement shall be compatible with the requested resource type. A client may set the value of the AllocationUnits property to NULL, requesting the implementation to assume a resource-type-specific default value for the unit of measurement.

An implementation shall use the value of the AllocationUnits property to determine the unit of measurement for the virtual resource allocation request. If the provided value is not compatible with the resource type, the implementation shall fail the method execution. If a value is not provided (NULL), the implementation shall assume a resource-type-specific default value for the unit of measurement. A resource-type-specific resource allocation DMTF management profile may specify rules for the determination of the default value.

5.2.7.9 CIM_ResourceAllocationSettingData.VirtualQuantity property

A client should set the value of the VirtualQuantity property to specify the quantity of virtual resources that shall result from the virtual resource allocation request. A client may set the value of the VirtualQuantity property to NULL, requesting a default behavior. A resource-type-specific resource allocation DMTF management profile may specify rules for the determination of a default value.

5.2.7.10 CIM_ResourceAllocationSettingData.Reservation property

A client may set the value of the Reservation property to specify the amount of host resource that is requested by the virtual resource allocation request. The unit of measurement established by the value of the AllocationUnits property applies. A client may set the value of the Reservation property to NULL, requesting a default behavior.

An implementation shall use the value of the Reservation property to determine the amount of host resource that is requested by the virtual resource allocation request. If a value is not provided (NULL), the implementation may exhibit an implementation-specific default behavior. A resource-type-specific resource allocation DMTF management profile may specify rules for the determination of a default value.

5.2.7.11 CIM_ResourceAllocationSettingData.Limit property

A client may set the value of the Limit property to specify the maximum amount of host resource that is requested by the virtual resource allocation request. The unit of measurement established by the value of the AllocationUnits property applies. A client may set the value of the Limit property to NULL, requesting a default behavior.

An implementation shall use the value of the Limit property to determine the maximum amount of host resource that is requested by the virtual resource allocation request. If a value is not provided (NULL), the implementation may exhibit an implementation-specific default behavior. A resource-type-specific resource allocation DMTF management profile may specify rules for the determination of a default value.

5.2.7.12 CIM_ResourceAllocationSettingData.Weight property

A client may set the value of the Weight property to specify a relative weight that is requested by the virtual resource allocation request with respect to other virtual resource allocation requests from the same resource pool. A client may set the value of the Weight property to NULL, requesting a default behavior.

An implementation shall use the value of the Weight property to determine a relative weight that is requested by the virtual resource allocation request with respect to other virtual resource allocation requests from the same resource pool. If a value is not provided (NULL), the implementation may exhibit an implementation-specific default behavior. A resource-type-specific resource allocation DMTF management profile may specify rules for the determination of a default value.

5.2.7.13 CIM_ResourceAllocationSettingData.AutomaticAllocation property

A client may set the value of the AutomaticAllocation property to specify that the requested resource allocation is obtained automatically when the virtual system is activated. A client may set the value of the AutomaticAllocation property to NULL, requesting a default value of TRUE.

An implementation shall use the value of the AutomaticAllocation property to determine whether the requested resource allocation is obtained automatically when the virtual system is activated. The default value shall be TRUE, requesting automatic resource allocation. Resource-type-specific resource allocation DMTF management profiles may specify a different default behavior.

5.2.7.14 CIM_ResourceAllocationSettingData.AutomaticDeallocation property

A client may set the value of the AutomaticDeallocation property to specify that the requested resource allocation is released automatically when the virtual system is deactivated. A client may set the value of the AutomaticDeallocation property to NULL, requesting a default value of TRUE.

An implementation shall use the value of the AutomaticDeallocation property to determine whether the resource allocation is automatically released when the virtual system is de-activated. The default value shall be TRUE, requesting automatic resource deallocation. Resource-type-specific resource allocation DMTF management profiles may specify a different default behavior.

5.2.7.15 CIM_ResourceAllocationSettingData.Parent property

A client may set the value of the Parent property to specify a parent resource required to establish the resource allocation. An example of such a parent resource would be a controller. A client may set the value of the Parent property to NULL, requesting a default behavior.

An implementation shall use the value of the Parent property to determine if and which parent resource is required for the requested resource allocation. If no value is specified (NULL), the implementation may decide whether a parent resource is needed and eventually select one itself. If a value is specified, the implementation shall base the resource allocation request on the requested parent resource. If the requested parent resource is not capable to support the requested resource allocation, the implementation shall fail the request.

5.2.7.16 CIM_ResourceAllocationSettingData.Connection[ ] array property

A client may set the value of the Connection[ ] array property to specify connection information for the requested resource allocation. An example of connection information is the target network for a network adaptor or the target switch port for storage adaptors. Connection information is highly dependent on resource type and implementation; for details, refer to the resource-type-specific resource allocation DMTF management profile.

An implementation shall use the values within the Connection[ ] array property to determine connection information for the requested resource allocation. If no value is specified (NULL), the implementation may decide whether the requested resource allocation requires connection information and establish a default connection.

5.2.7.17 CIM_ResourceAllocationSettingData.Address property

A client may set the value of the Address property to specify an address for the new virtual device. In general, the requirement for the value of the Address property will depend on the resource type. For a particular resource type, restrictions on the potential value set may exist.

An implementation shall interpret the value of the Address property such that the new virtual resource adopts that address value while it is instantiated. If no value is specified (NULL), the implementation may assign a value for the Address that is specific to the implementation and resource type.

5.2.7.18 CIM_ResourceAllocationSettingData.MappingBehavior property

A client may set the value of the MappingBehavior property to specify whether the requested resource allocation has an affinity to or is directly based on host resources that are specified in the optional HostResource[ ] array property (see 5.2.7.7). A client may set the value of the MappingBehavior property to NULL or to 0 (Unknown) to request that the implementation shall decide on the mapping behavior. A client shall not specify a value of 1 (Not Supported).

An implementation shall use the value of the MappingBehavior property to determine how the requested resource allocation depends on host resources that are specified in the HostResource[ ] array property.

- If the value is 1 (Not Supported), the implementation shall return a return code that indicates an invalid parameter; see 5.3.
- If the value is not provided (NULL), is 0 (Unknown), or is 1 (Not Supported), the implementation shall establish an implementation-specific default behavior. The resource request may or may not be mapped to or based on host resources depending on the implementation's decision.
- If the value is 2 (Dedicated), the implementation shall establish a direct mapping of the virtual resource onto the resources specified through the HostResource[ ] array property. The implementation may establish a mapping in the resource allocation request resulting from this instance of the CIM_ResourceAllocationSettingData class; however, it is possible that the requested resources are not available at resource allocation time, resulting in an error condition at that time.

5.3 Methods

This subclause details the requirements for supporting intrinsic operations and extrinsic methods for the CIM elements defined by the profile described in this clause.

5.3.1 CIM_ResourcePoolConfigurationService.CreateChildResourcePool( )

The CIM Schema description of this method applies. This optional method creates (or starts a job to create) a nested resource pool. Refer to the MOF for a detailed description.

If the SupportedSyncMethods property of the associated instance of CIM_ResourcePoolConfigurationCapabilities is set to 3 (CreateChildResourcePool Is Supported), the CreateChildResourcePool( ) method shall be implemented and shall not return a value of 1 or 4096.

If the SupportedAsyncMethods property of the associated instance of CIM_ResourcePoolConfigurationCapabilities is set to 3 (CreateChildResourcePool Is Supported), the CreateChildResourcePool( ) method shall be implemented and shall not return a value of 1.

If neither the SupportedSyncMethods nor the SupportedAsyncMethods property of the associated instance of CIM_ResourcePoolConfigurationCapabilities is set to 3 (CreateChildResourcePool Is Supported), the CreateChildResourcePool( ) method shall not be implemented, or if the method is implemented, it shall return the value 1.

If an implementation supports the resource pool hierarchy device model, it uses this method to create child pools.

Detailed requirements of the CreateChildResourcePool( ) method are specified in Table 3 and Table 4.

No standard messages are defined.

Table 3 – CIM_ResourcePoolConfigurationService.CreateChildResourcePool( ) method: Return code values

Value	Description
0	Job completed with no error
1	Not supported
2	Unknown
3	Timeout
4	Failed
5	Invalid parameter
6	In use
7	Incorrect ResourceType for the pool
8	Insufficient resources
4096	Method parameters checked – job started

Table 4 – CIM_ResourcePoolConfigurationService.CreateChildResourcePool( ) method: Parameters

Qualifiers	Name	Type	Description/Values
IN	ElementName	String	The desired name of the resource pool
IN	Settings	String	A string representation of a CIM_ResourceAllocationSettingData instance that represents the allocation assigned to this child pool
IN	ParentPool	CIM_ResourcePool REF	The parent pool from which to create this pool
OUT	Pool	CIM_ResourcePool REF	The resulting resource pool
OUT	Job	CIM_ConcreteJob REF	Returned job if started
OUT	Error	String	Encoded error instance if the operation failed and did not return a job

5.3.2 CIM_ResourcePoolConfigurationService.DeleteResourcePool( )

The CIM Schema description of this method applies. This optional method deletes (or starts a job to delete) a resource pool. Refer to the MOF for a detailed description.

If the SupportedSyncMethods property of the associated instance of CIM_ResourcePoolConfigurationCapabilities is set to 4 (DeleteResourcePool Is Supported), the DeleteResourcePool( ) method shall be implemented and shall not return a value of 1 or 4096.

If the SupportedAsyncMethods property of the associated instance of CIM_ResourcePoolConfigurationCapabilities is set to 4 (DeleteResourcePool Is Supported), the DeleteResourcePool( ) method shall be implemented and shall not return a value of 1.

If neither the SupportedSyncMethods nor the SupportedAsyncMethods property of the associated instance of CIM_ResourcePoolConfigurationCapabilities is set to 4 (DeleteResourcePool Is Supported), the DeleteResourcePool( ) method shall not be implemented, or if the method is implemented, it shall return the value 1.

Detailed requirements of the DeleteResourcePool( ) method are specified in Table 5 and Table 6.

No standard messages are defined.

Table 5 – CIM_ResourcePoolConfigurationService.DeleteResourcePool( ) method: Return code values

Value	Description
0	Job completed with no error
1	Not supported
2	Unknown
3	Timeout
4	Failed
5	Invalid parameter
6	In use
7	Incorrect ResourceType for the pool
4096	Method parameters checked – job started

Table 6 – CIM_ResourcePoolConfigurationService.DeleteResourcePool( ) method: Parameters

Qualifiers	Name	Type	Description/Values
IN	Pool	CIM_ResourcePool REF	The resource pool to delete
OUT	Job	CIM_ConcreteJob REF	Returned job if started
OUT	Error	String	Encoded error instance if the operation failed and did not return a job

5.3.3 CIM_ResourcePoolConfigurationService.AddResourcesToResourcePool( )

The CIM Schema description of this method applies. This optional method adds (or starts a job to add) resources to a resource pool. Refer to the MOF for a detailed description.

If the SupportedSyncMethods property of the associated instance of CIM_ResourcePoolConfigurationCapabilities is set to 5 (AddResourcesToResourcePool Is Supported), the AddResourcesToResourcePool( ) method shall be implemented and shall not return a value of 1 or 4096.

If the SupportedAsyncMethods property of the associated instance of CIM_ResourcePoolConfigurationCapabilities is set to 5 (AddResourcesToResourcePool Is Supported), the AddResourcesToResourcePool( ) method shall be implemented and shall not return a value of 1.

If neither the SupportedSyncMethods nor the SupportedAsyncMethods property of the associated instance of CIM_ResourcePoolConfigurationCapabilities is set to 5 (AddResourcesToResourcePool Is Supported), the AddResourcesToResourcePool( ) method shall not be implemented, or if the method is implemented, it shall return the value 1.

Detailed requirements of the AddResourcesToResourcePool( ) method are specified in Table 7 and Table 8.

No standard messages are defined.

Table 7 – CIM_ResourcePoolConfigurationService.AddResourcesToResourcePool( ) method: Return code values

Value	Description
0	Job completed with no error
1	Not supported
2	Unknown
3	Timeout
4	Failed
5	Invalid parameter
6	In use
7	Incorrect ResourceType for the pool
4096	Method parameters checked – job started

Table 8 – CIM_ResourcePoolConfigurationService.AddResourcesToResourcePool( ) method: Parameters

Qualifiers	Name	Type	Description/Values
IN	HostResource[ ]	CIM_LogicalDevice REF[ ]	The host resources to assign to the pool
IN	Pool	CIM_ResourcePool REF	The primordial ResourcePool to add resources to
OUT	Job	CIM_ConcreteJob REF	Returned job if started
OUT	Error	String	Encoded error instance if the operation failed and did not return a job

5.3.4 CIM_ResourcePoolConfigurationService.RemoveResourcesFromResource Pool( )

The CIM Schema description of this method applies. This optional method removes (or starts a job to remove) resources from a resource pool. Refer to the MOF for a detailed description.

If the SynchronousMethodsSupported property of the associated instance of CIM_ResourcePoolConfigurationCapabilities is set to 6 (RemoveResourcesFromResourcePool Is Supported), the RemoveResourcesFromResourcePool( ) method shall be implemented and shall not return a value of 1 or 4096.

If the AsynchronousMethodsSupported property of the associated instance of CIM_ResourcePoolConfigurationCapabilities is set to 6 (RemoveResourcesFromResourcePool Is Supported), the RemoveResourcesFromResourcePool( ) method shall be implemented and shall not return a value of 1.

If neither the SynchronousMethodsSupported nor the AsynchronousMethodsSupported property of the associated instance of CIM_ResourcePoolConfigurationCapabilities is set to 6 (RemoveResourcesFromResourcePool Is Supported), the RemoveResourcesFromResourcePool( ) method shall not be implemented, or if the method is implemented, it shall return the value 1.

Detailed requirements of the RemoveResourcesFromResourcePool( ) method are specified in Table 9 and Table 10.

No standard messages are defined.

Table 9 – CIM_ResourcePoolConfigurationService.RemoveResourcesFromResourcePool( ) method: Return code values

Value	Description
0	Job completed with no error
1	Not supported
2	Unknown
3	Timeout
4	Failed
5	Invalid parameter
6	In use
7	Incorrect ResourceType for the pool
8	Insufficient resources
4096	Method parameters checked – job started

Table 10 – CIM_ResourcePoolConfigurationService.RemoveResourcesFromResourcePool( ) method: Parameters

Qualifiers	Name	Type	Description/Values
IN	HostResource[ ]	CIM_LogicalDevice REF[ ]	The host resources to remove from the pool
IN	Pool	CIM_ResourcePool REF	The primordial ResourcePool to remove resources from
OUT	Job	CIM_ConcreteJob REF	Returned job if started
OUT	Error	String	Encoded error instance if the operation failed and did not return a job

5.3.5 CIM_ResourcePoolConfigurationService.ChangeParentResourcePool( )

The CIM Schema description of this method applies. This optional method changes (or starts a job to change) a parent resource pool. Refer to the MOF for a detailed description.

If the SupportedSyncMethods property of the associated instance of CIM_ResourcePoolConfigurationCapabilities is set to 7 (ChangeParentResourcePool Is Supported), the ChangeParentResourcePool( ) method shall be implemented and shall not return a value of 1 or 4096.

If the SupportedAsyncMethods property of the associated instance of CIM_ResourcePoolConfigurationCapabilities is set to 7 (ChangeParentResourcePool Is Supported), the ChangeParentResourcePool( ) method shall be implemented and shall not return a value of 1.

If neither the SupportedSyncMethods nor the SupportedAsyncMethods property of the associated instance of CIM_ResourcePoolConfigurationCapabilities is set to 7 (ChangeParentResourcePool Is Supported), the ChangeParentResourcePool( ) method shall not be implemented, or if the method is implemented, it shall return the value 1.

Detailed requirements of the ChangeParentResourcePool( ) method are specified in Table 11 and Table 12.

No standard messages are defined.

Table 11 – CIM_ResourcePoolConfigurationService.ChangeParentResourcePool( ) method: Return code values

Value	Description
0	Job completed with no error
1	Not supported
2	Unknown
3	Timeout
4	Failed
5	Invalid parameter
6	In use
7	Incorrect ResourceType for the pool
8	Insufficient resources
4096	Method parameters checked – job started

Table 12 – CIM_ResourcePoolConfigurationService.ChangeParentResourcePool( ) method: Parameters

Qualifiers	Name	Type	Description/Values
IN	ParentPool	CIM_ResourcePool REF	The parent resource pool to change to
IN	Settings	String	A string representation of an instance of CIM_ResourceAllocationSettingData that represents the allocation assigned to this child pool
OUT	Job	CIM_ConcreteJob REF	Returned job if started
OUT	Error	String	Encoded error instance if the operation failed and did not return a job

5.3.6 Profile conventions for operations

For each profile class (including associations), the implementation requirements for operations, including those in the following default list, are specified in class-specific subclauses of this subclause.

The default list of operations for all classes is:

- GetInstance( )
- EnumerateInstances( )
- EnumerateInstanceNames( )

For classes that are referenced by an association, the default list also includes

- Associators( )
- AssociatorNames( )
- References( )
- ReferenceNames( )

5.3.7 CIM_AffectedJobElement

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.8 CIM_BaseMetricDefinition

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.9 CIM_BaseMetricValue

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.10 CIM_Component

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.11 CIM_ConcreteJob

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.12 CIM_ElementAllocatedFromPool

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.13 CIM_ElementCapabilities

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.14 CIM_ElementSettingData

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.15 CIM_HostedDependency

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.16 CIM_HostedResourcePool

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.17 CIM_HostedService

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.18 CIM_LogicalDevice

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.19 CIM_MetricDefForME

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.20 CIM_MetricForME

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.21 CIM_MetricInstance

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.22 CIM_ResourceAllocationFromPool

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.23 CIM_ResourceAllocationSettingData

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.24 CIM_ResourcePool

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.25 CIM_ResourcePoolConfigurationCapabilities

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.26 CIM_ResourcePoolConfigurationService

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.27 CIM_ServiceAffectsElement

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.3.28 CIM_SystemDevice

All operations in the default list in 5.3.6 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

5.4 Use cases

This subclause contains object diagrams and use cases that represent the intended use of the profile described in this clause. The use cases are informative and not intended to define the requirements for conformance.

5.4.1 Abstract instance diagram

Figure 2 illustrates the use of the Resource Allocation Profile with a primordial pool, a concrete pool, and backed resources used for virtualization.

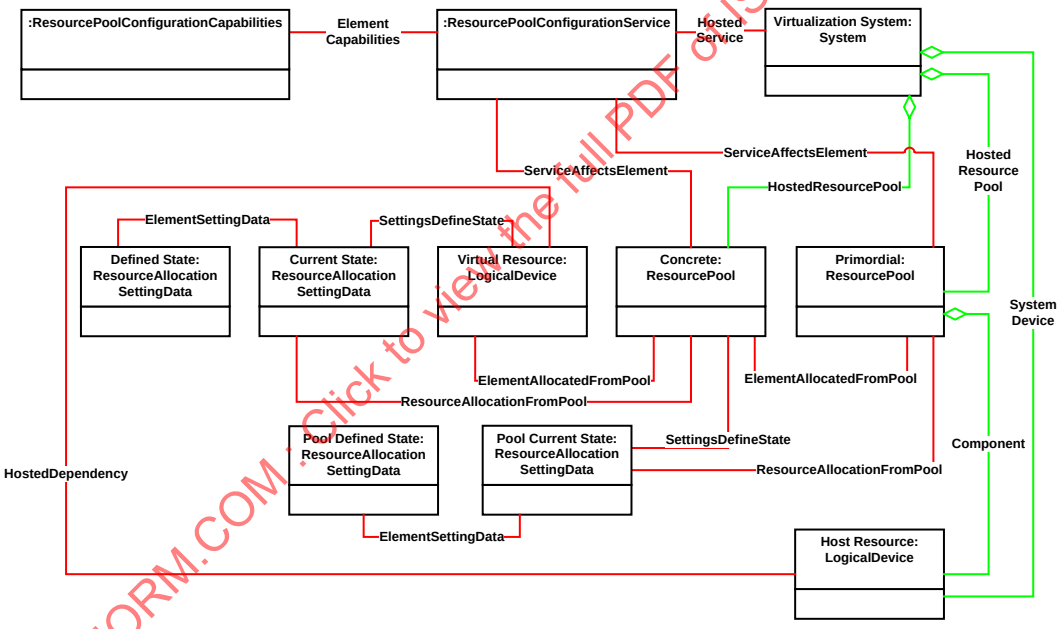

Figure 2 – Abstract instance diagram: Concrete resource pool

Figure 3 illustrates the use of the Resource Allocation Profile with a primordial pool and backed resources used for virtualization. Resources are allocated directly from the primordial pool to consumers.

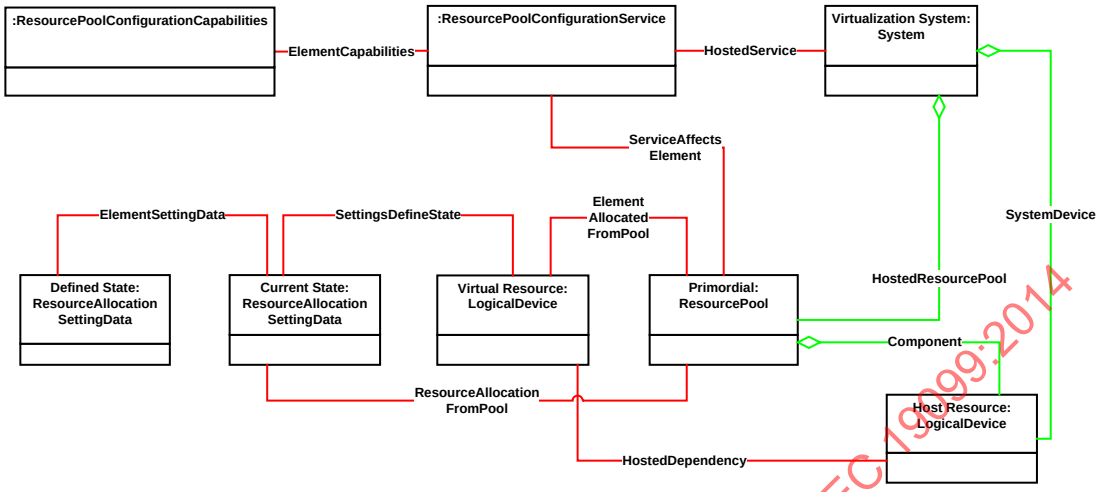

Figure 3 – Abstract instance diagram: Primordial pool with backed resources

Figure 4 illustrates the use of the Resource Allocation Profile with a primordial pool that does not have backed resources used for virtualization. The resources are either synthetic (that is, no physical elements are backing them) or not modeled by the implementation.

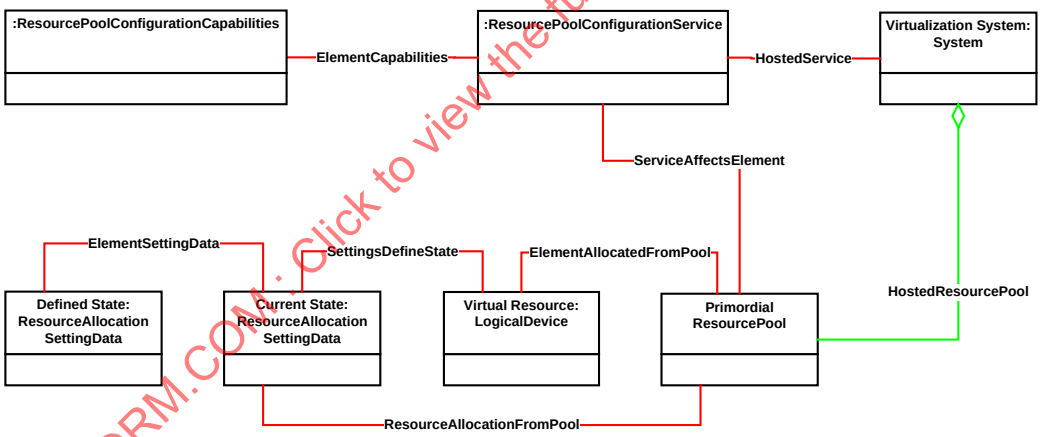

Figure 4 – Abstract instance diagram: Primordial pool without backed resources

5.4.2 Resource pool hierarchy diagram

Figure 5 shows a hierarchy of related resource pools in which host resources are shared. Child resource pools are allocated from a parent resource pool by using the same pattern as virtual resources. The host resources are members of the top-most or primordial resource pool. An instance of CIM_ResourceAllocationSettingData for a descendant resource pool records the way resources flow from the parent resource pool to the child resource pool. For example, if only weight is set, the child resource pool shares all resources with other child resource pools that have allocations scheduled based on the weight of the child resource pool.

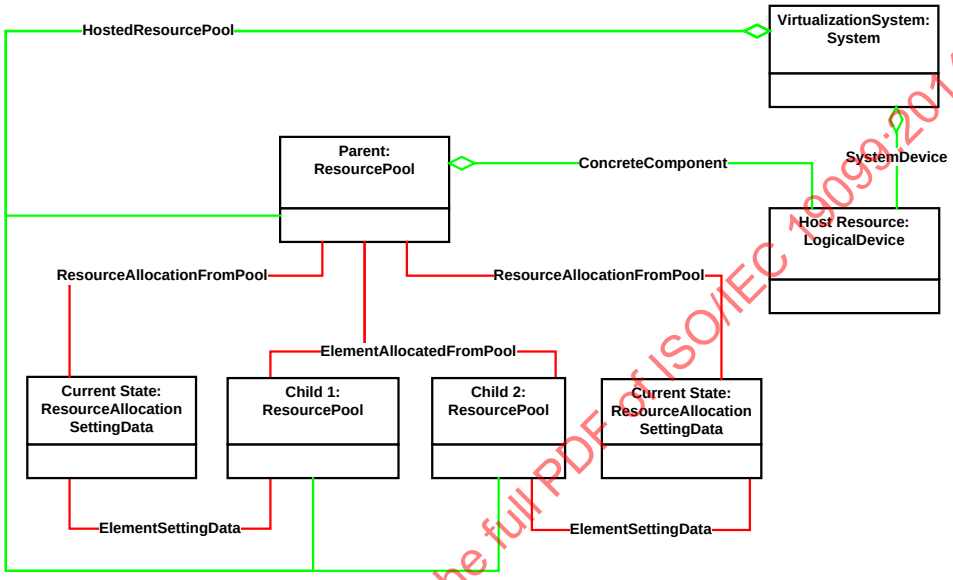

Figure 5 – Resource pool hierarchy instance diagram

5.4.3 Simple resource allocation diagram

Figure 6 shows Simple Resource Allocation. Two instances of a subclass of CIM_LogicalDevice are aggregated into the CIM_ResourcePool instance through the CIM_ConcreteComponent association. This indicates that the components modeled by the device contribute resources into a pool from which the resources may be allocated.

CIM_LogicalDevice is shown to indicate that numerous different component types may be aggregated into the pool. rasd2 and rasd1 represent allocations from the pool for two resource consumers represented by Mse2 and Mse1. These allocations are indicated by the CIM_ResourceAllocationFromPool associations between rasd1 and rasd2 and the CIM_ResourcePool instance, and the CIM_ElementSettingData associations between the CIM_ResourceAllocationSettingData instances and the CIM_ManagedSystemElement instances.

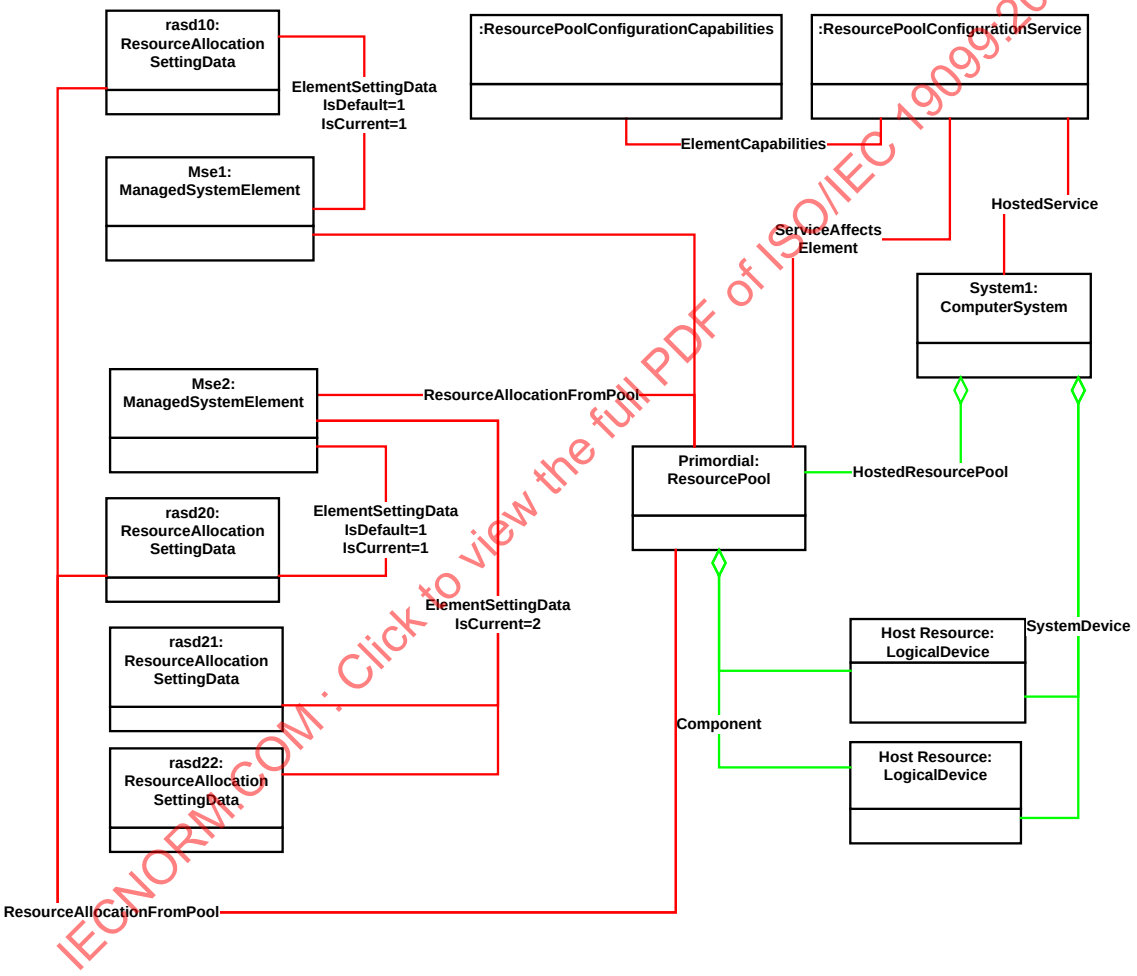

Figure 6 – Simple resource allocation

5.4.4 Determine pool type

A client may determine the type of resource provided by the resource pool by querying the ResourceType property in an instance of CIM_ResourcePool.

5.4.5 View historical use of pool resource by a resource consumer

The historical use of a resource by a particular resource consumer may be useful for reporting or billing purposes. It may also be useful for planning purposes in order to predict future use of the resource. A client may determine the historical use of a resource by a consumer as follows:

1) If device resource allocation is implemented, find all instances of CIM_LogicalDevice that are associated with the resource consumer of the type that corresponds to the resource type. Use each CIM_LogicalDevice instance as the subject instance in step 3) and sum the results.
2) If simple resource allocation is implemented, use the CIM_ManagedSystemElement instance that represents the resource consumer as the subject instance in step 3).
3) Find an instance of CIM_BaseMetricDefinition, as defined in 5.5.2.3, that is associated with the subject instance through the CIM_MetricDefForME association.
4) Find all instances of CIM_BaseMetricValue that are associated with the CIM_BaseMetricDefinition through the CIM_MetricInstance association where the CIM_BaseMetricValue.BreakdownValue property has the value of the PoolID property of the CIM_ResourcePool instance of interest.

5.4.6 View historical aggregate use of a pool resource

The historical aggregate use of resources in the pool may be useful in predicting future resource requirements. A client may determine the historical aggregate use of a resource by a consumer as follows:

1) Find an instance of CIM_BaseMetricDefinition, as defined in 5.5.2.3, that is associated with the CIM_ResourcePool instance through the CIM_MetricDefForME association.
2) Find all instances of CIM_BaseMetricValue that are associated with the CIM_BaseMetricDefinition through the CIM_MetricInstance association where the CIM_BaseMetricValue.BreakdownValue property has the value of the PoolID property of the CIM_ResourcePool instance of interest.

5.4.7 Discover host resources

A client may find all the host resources of a specific type as follows:

1) Find instances of CIM_ResourcePool with the Primordial property set to TRUE and the ResourceType property set as desired.
2) Find all instances of CIM_ManagedSystemElement that are associated with the CIM_ResourcePool instances through the CIM_ConcreteComponent association.

5.4.8 Discover supported resource types

A client may find all the resource types supported by the allocation platform as follows:

3) Enumerate resource pools and find primordial pool types.
4) Identify the unique ResourceType property values within the list of CIM_ResourcePool instances.

5.5 CIM elements

Table 13 lists CIM elements that are specified or specialized for the profile described in this clause. Each CIM element shall be implemented as described in Table 13. Subclauses 5.2 (“Implementation”) and 5.3 (“Methods”) may impose additional requirements on these elements.

Table 13 – CIM elements: Resource Allocation Profile

Element name	Requirement	Description
Classes		
CIM_AffectedJobElement	Optional	See 5.5.1.
CIM_BaseMetricDefinition	Optional	See 5.5.2, 5.5.2.1, 5.5.2.2, and 5.5.2.3.
CIM_BaseMetricValue	Optional	See 5.5.3, 5.5.3.1, 5.5.3.2, and 5.5.3.3.
CIM_Component	Conditional	See 5.5.4.
CIM_ConcreteJob	Optional	See 5.5.5.
CIM_ElementAllocatedFromPool	Mandatory	See 5.5.6.
CIM_ElementCapabilities	Mandatory	See 5.5.7.
CIM_ElementSettingData	Mandatory	See 5.5.8.
CIM_HostedResourcePool	Mandatory	See 5.5.10.
CIM_HostedService	Mandatory	See 5.5.11.
CIM_LogicalDevice (virtual resource)	Mandatory	See 5.5.12.
CIM_MetricDefForME	Conditional	See 5.5.13.
CIM_MetricForME	Conditional	See 5.5.14.
CIM_ResourceAllocationFromPool	Optional	See 5.5.15.
CIM_ResourceAllocationSettingData	Conditional	See 5.5.16 and 5.5.17.
CIM_ResourcePool	Mandatory	See 5.5.18.
CIM_ResourcePoolConfigurationCapabilities	Mandatory	See 5.5.19.
CIM_ResourcePoolConfigurationService	Mandatory	See 5.5.20.
CIM_SettingsDefineState	Mandatory	See 5.5.21.
CIM_ServiceAffectsElement	Mandatory	See 5.5.22.
CIM_SystemDevice	Conditional	See 5.5.23.
CIM_HostedDependency	Optional	See 5.5.9.
Indications		
None defined in the profile described in this clause		

5.5.1 CIM_AffectedJobElement

If long-running jobs are supported, this association provides a reference to the affected element. For example, if a new CIM_ResourcePool instance is created and a CIM_ConcreteJob instance is returned, after that CIM_ConcreteJob instance indicates that the create operation has completed the CIM_AffectedJobElement association may be used to locate the resulting CIM_ResourcePool instance. Table 14 defines the requirements for elements of this class.

Table 14 – Class: CIM_AffectedJobElement

Elements	Requirement	Notes
AffectedElement	Mandatory	The affected element (for example, the CIM_ResourcePool) Cardinality 1
AffectingElement	Mandatory	The CIM_ConcreteJob Cardinality 1

5.5.2 CIM_BaseMetricDefinition

CIM_BaseMetricDefinition defines metrics that are maintained for the resource pool and resource consumers. Table 15 defines the requirements for elements of this class.

Table 15 – Class: CIM_BaseMetricDefinition

Elements	Requirement	Notes
BreakdownDimensions	Mandatory	Matches ("CIM_ResourcePool.PoolID")
Calculatable	Mandatory	None
ChangeType	Mandatory	None
DataType	Mandatory	None
ElementName	Mandatory	Pattern (".+").
GatheringType	Mandatory	None
ID	Mandatory	**Key**
IsContinuous	Mandatory	None
TimeScope	Mandatory	None
TimeScope	Optional	None
Units	Mandatory	None

5.5.2.1 CIM_BaseMetricDefinition — Instantaneous consumption

CIM_BaseMetricDefinition defines metrics that are maintained for the resource pool and resource consumers. Table 16 describes the requirements for using CIM_BaseMetricDefinition to define the metric for instantaneous consumption. These constraints are in addition to those specified in 5.5.2.

Table 16 – Class: CIM_BaseMetricDefinition — Instantaneous consumption

Elements	Requirement	Notes
Calculatable	Mandatory	Matches 3 (Non-summable)
ChangeType	Mandatory	Matches 4 (Gauge)
DataType	Mandatory	Matches 13 (uint64)
ElementName	Mandatory	Pattern (".+")
TimeScope	Mandatory	Matches (Point)

5.5.2.2 CIM_BaseMetricDefinition — Interval metrics

CIM_BaseMetricDefinition defines metrics that are maintained for the resource pool and resource consumers. Table 17 describes the requirements for using CIM_BaseMetricDefinition to define the metric for interval metrics. These constraints are in addition to those specified in 5.5.2.

Table 17 – Class: CIM_BaseMetricDefinition — Interval metrics

Elements	Requirement	Notes
Calculatable	Mandatory	Matches 2 (Summable)
ChangeType	Mandatory	Matches 4 (Gauge)
DataType	Mandatory	Matches 13 (uint64)
TimeScope	Mandatory	Matches 3 (Interval)

5.5.2.3 CIM_BaseMetricDefinition — Aggregate consumption

CIM_BaseMetricDefinition defines metrics that are maintained for the resource pool and resource consumers. Table 18 describes the requirements for using CIM_BaseMetricDefinition to define the metric for aggregate consumption. These constraints are in addition to those specified in 5.5.2.

Table 18 – Class: CIM_BaseMetricDefinition — Aggregate consumption

Elements	Requirement	Notes
Calculatable	Mandatory	Matches 3 (Non-summable)
ChangeType	Mandatory	Matches 3 (Counter)
DataType	Mandatory	Matches 13 (uint64)
TimeScope	Mandatory	Matches 3 (Interval)

5.5.3 CIM_BaseMetricValue

CIM_BaseMetricValue conveys the actual defined data of a metric that has been maintained for a resource pool or resource consumer. Table 19 defines the requirements for elements of this class.

Table 19 – Class: CIM_BaseMetricValue

Elements	Requirement	Notes
MetricDefinitionID	Mandatory	None
MetricValue	Mandatory	None
Duration	Optional	None
TimeStamp	Optional	None
Volatile	Mandatory	None
InstanceID	Mandatory	**Key**
BreakdownDimension	Mandatory	Matches ("CIM_ResourcePool.PoolID")
BreakdownValue	Mandatory	Shall match the value of the CIM_ResourcePool.PoolID property for the pool from which the resource was consumed

5.5.3.1 CIM_BaseMetricValue — Instantaneous consumption

CIM_BaseMetricValue reports a metric that is defined using CIM_BaseMetricDefinition. Table 20 describes the requirements for using CIM_BaseMetricValue to report the metric for instantaneous consumption. These constraints are in addition to those specified in 5.5.3.

Table 20 – Class: CIM_BaseMetricValue — Instantaneous consumption

Elements	Requirement	Notes
Duration	Mandatory	None
Timestamp	Mandatory	None
Volatile	Mandatory	Matches TRUE

5.5.3.2 CIM_BaseMetricValue — Interval metrics

CIM_BaseMetricValue reports a metric that is defined using CIM_BaseMetricDefinition. Table 21 defines the requirements for using CIM_BaseMetricValue to report the metric for interval metrics. These constraints are in addition to those specified in 5.5.3.

Table 21 – Class: CIM_BaseMetricValue — Interval metrics

Elements	Requirement	Notes
Duration	Mandatory	None
Timestamp	Mandatory	None
Volatile	Mandatory	Matches TRUE

5.5.3.3 CIM_BaseMetricValue — Aggregate consumption

CIM_BaseMetricValue reports a metric that is defined using CIM_BaseMetricDefinition. Table 22 defines the requirements for using CIM_BaseMetricValue to report the metric for aggregate consumption. These constraints are in addition to those specified in 5.5.3.

Table 22 – Class: CIM_BaseMetricValue — Aggregate consumption

Elements	Requirement	Notes
Duration	Mandatory	None
Timestamp	Mandatory	None
Volatile	Mandatory	Matches TRUE

5.5.4 CIM_Component

CIM_Component associates a host resource with the resource pool. Table 23 defines the requirements for elements of this class.

Table 23 – Class: CIM_Component

Elements	Requirement	Notes
PartComponent	Mandatory	Shall be a reference to an instance of CIM_ManagedElement that represents a Host Resource Cardinality *
GroupComponent	Mandatory	Shall be a reference to an instance of CIM_ResourcePool Cardinality 0..1

5.5.5 CIM_ConcreteJob

CIM_ConcreteJob is used to manage the results of long-running operations to manage resource pools. Table 24 defines the requirements for elements of this class.

Table 24 – Class: CIM_ConcreteJob

Elements	Requirement	Notes
ElementName	Mandatory	(pattern ".*")
InstanceID	Mandatory	None
JobState	Mandatory	None
DeleteOnCompletion	Mandatory	Matches TRUE
ErrorCode	Mandatory	None
ErrorDescription	Mandatory	None
JobStatus	Mandatory	None
TimeBeforeRemoval	Mandatory	None

5.5.6 CIM_ElementAllocatedFromPool

CIM_ElementAllocatedFromPool is used to associate a CIM_LogicalElement that represents a virtual resource or child pool with the CIM_ResourcePool instance from which the resource was allocated. Table 25 defines the requirements for elements of this class.

Table 25 – Class: CIM_ElementAllocatedFromPool

Elements	Requirement	Notes
Antecedent	Mandatory	Shall be a reference to an instance of CIM_ResourcePool Cardinality 1
Dependent	Mandatory	Shall be a reference to an instance of a subclass of CIM_LogicalElement that represents the virtual resource or child pool Cardinality *

5.5.7 CIM_ElementCapabilities

CIM_ElementCapabilities associates a resource pool configuration service to the capabilities instance that describes the methods supported by the service. Table 26 defines the requirements for elements of this class.

Table 26 – Class: CIM_ElementCapabilities

Elements	Requirement	Notes
Capabilities	Mandatory	Shall be a reference to an instance of CIM_ResourcePoolConfigurationCapabilities Cardinality 1
ManagedElement	Mandatory	Shall be a reference to an instance of CIM_ResourcePoolConfigurationService Cardinality 1..*

5.5.8 CIM_ElementSettingData

The CIM_ElementSettingData association shall be used to associate an instance of the CIM_SettingData class that represents a resource allocation as part of the resource allocation state with corresponding instances of the CIM_ResourceAllocationSettingData class that describe the same allocation element for the virtual resource in a different context, such as, for example, the resource allocation definition. Table 27 defines the requirements for elements of this class.

Table 27 – Class: CIM_ElementSettingData

Elements	Requirement	Notes
ManagedElement	Mandatory	Shall be a reference to an Allocation Target Cardinality *
SettingData	Mandatory	Shall be a reference to an instance of CIM_ResourceAllocationSettingData Cardinality 1..*

5.5.9 CIM_HostedDependency

CIM_HostedDependency associates a virtual resource with a specific host resource. Table 28 defines the requirements for elements of this class.

Table 28 – Class: CIM_HostedDependency

Elements	Requirement	Notes
Antecedent	Mandatory	Shall be a reference to an instance of CIM_LogicalDevice that represents a Host Resource Cardinality 0..1
Dependent	Mandatory	Shall be a reference to an instance of CIM_LogicalDevice that represents a Virtual Resource Cardinality 1

5.5.10 CIM_HostedResourcePool

CIM_HostedResourcePool associates a resource pool with a hosting system. Table 29 defines the requirements for elements of this class.

Table 29 – Class: CIM_HostedResourcePool

Elements	Requirement	Notes
Antecedent	Mandatory	Shall be a reference to the Host Instance Cardinality 1
Dependent	Mandatory	Shall be a reference to the Central Instance Cardinality 1..*

5.5.11 CIM_HostedService

CIM_HostedService associates a CIM_ResourcePoolConfigurationService with a host system. Table 30 defines the requirements for elements of this class.

Table 30 – Class: CIM_HostedService

Elements	Requirement	Notes
Antecedent	Mandatory	Shall be a reference to an instance of CIM_System Cardinality 1
Dependent	Mandatory	Shall be a reference to an instance of CIM_ResourcePoolConfigurationService Cardinality *

5.5.12 CIM_LogicalDevice (virtual resource)

CIM_LogicalDevice is used to represent a virtual resource. Table 31 defines the requirements for elements of this class.

Table 31 – Class: CIM_LogicalDevice

Elements	Requirement	Notes
SystemCreationClassName	Mandatory	**Key**
SystemName	Mandatory	**Key**
CreationClassName	Mandatory	**Key**
DeviceID	Mandatory	**Key**

5.5.13 CIM_MetricDefForME

CIM_MetricForME relates a metric to the managed element for which it was measured. Table 32 defines the requirements for elements of this class.

Table 32 – Class: CIM_MetricDefForME

Elements	Requirement	Notes
Antecedent	Mandatory	Shall be a reference to an instance of CIM_ManagedElement Cardinality 1
Dependent	Mandatory	Shall be a reference to CIM_BaseMetricDefinition Cardinality *

5.5.14 CIM_MetricForME

CIM_MetricForME relates a metric to the managed element for which it was measured. Table 33 defines the requirements for elements of this class.

Table 33 – Class: CIM_MetricForME

Elements	Requirement	Notes
Antecedent	Mandatory	Shall be a reference to an instance of CIM_ManagedElement Cardinality 1
Dependent	Mandatory	Shall be a reference to CIM_BaseMetricInstance Cardinality *

5.5.15 CIM_ResourceAllocationFromPool

CIM_ResourceAllocationFromPool is used to associate an instance of CIM_ResourceAllocationSettingData with the CIM_ResourcePool instance from which the resource was allocated. Table 34 defines the requirements for elements of this class.

Table 34 – Class: CIM_ResourceAllocationFromPool

Elements	Requirement	Notes
Antecedent	Mandatory	Shall be a reference to an instance of CIM_ResourcePool class that represents a resource pool. Cardinality 0..1
Dependent	Mandatory	Shall be a reference to an instance of CIM_ResourceAllocationSettingData. Cardinality *

5.5.16 CIM_ResourceAllocationSettingData (Resource Allocation)

An instance of class CIM_ResourceAllocationSettingData shall be used to represent a resource allocation. If a virtualized resource is allocated and its CIM_LogicalDevice subclass instance is present, an instance of this class shall also be present to reflect the current virtual resource allocation settings. Table 35 defines the requirements for elements of this class.

Table 35 – Class: CIM_ResourceAllocationSettingData (current settings)

Elements	Requirement	Notes
Address	Optional	None
AllocationUnits	Mandatory	None
AutomaticAllocation	Optional	None
AutomaticDeallocation	Optional	None
Connection	Optional	None
HostResource[ ]	Optional	None
InstanceID	Mandatory	Opaque
IsVirtualized	Optional	None
Limit	Optional	None
MappingBehavior	Conditional	See 5.2.2.4 and 5.2.2.5.
OtherResourceType	Optional	None
Parent	Optional	None
PoolID	Mandatory	None
Reservation	Optional	None
ResourceSubType	Optional	None
ResourceType	Mandatory	None
VirtualQuantity	Optional	None

Elements	Requirement	Notes
Weight	Optional	None

5.5.17 CIM_ResourceAllocationSettingData (resource allocation request)

An instance of the CIM_ResourceAllocationSettingData class shall be used to represent a resource allocation request. Implementations may choose to use one instance to reflect both defined and current settings and point to references within the CIM_ElementSettingData association and the CIM_SettingsDefineState association, respectively. Table 36 defines the requirements for elements of this class.

Table 36 – Class: CIM_ResourceAllocationSettingData (defined settings)

Elements	Requirement	Notes
Address	Optional	None
AllocationUnits	Mandatory	None
AutomaticAllocation	Optional	None
AutomaticDeallocation	Optional	None
Connection	Optional	None
HostResource[ ]	Optional	None
InstanceID	Mandatory	Opaque
IsVirtualized	Optional	None
Limit	Optional	None
MappingBehavior	Conditional	See 5.2.2.4 and 5.2.2.5.
OtherResourceType	Optional	None
Parent	Optional	None
PoolID	Optional	None
Reservation	Optional	None
ResourceSubType	Optional	None
ResourceType	Mandatory	None
VirtualQuantity	Optional	None
Weight	Optional	None

5.5.18 CIM_ResourcePool

One or more CIM_ResourcePool instances may exist on a system for any given CIM_ResourceType instance. Table 37 defines the requirements for elements of this class.

Table 37 – Class: CIM_ResourcePool

Elements	Requirement	Notes
InstanceID	Mandatory	Opaque
PoolID	Mandatory	Opaque
Primordial	Mandatory	See 5.2.1.2.
Capacity	Conditional	See 5.2.1.2 and 5.2.2.1.

Elements	Requirement	Notes
Reserved	Optional	None
ResourceType	Mandatory	None
OtherResourceType	Optional	None
ResourceSubType	Optional	None
AllocationUnits	Conditional	Condition: Reserved or Capacity is implemented
ElementName	Optional	Pattern (".+")

5.5.19 CIM_ResourcePoolConfigurationCapabilities

All implementations shall implement this capabilities class, setting the supported properties to reflect the individual CIM_ResourcePoolConfigurationService methods supported by the implementation. Implementations of the individual service methods shall be either synchronous or asynchronous, but not both. Synchronous implementations may return quickly or slowly, and shall never return a Job. Asynchronous implementations shall always return quickly. If the operation is long running, the implementation shall return a Job to track the operation. Table 38 defines the requirements for elements of this class.

Table 38 – Class: CIM_ResourcePoolConfigurationCapabilities

Elements	Requirement	Notes
AsynchronousMethodsSupported	Mandatory	None
SynchronousMethodsSupported	Mandatory	None

5.5.20 CIM_ResourcePoolConfigurationService

The CIM_ResourcePoolConfigurationService provides for active management of Resource Pools. It allows jobs to be started for the creation and deletion of ResourcePools as well as addition and subtraction of host resources from ResourcePools. Table 39 defines the requirements for the CIM_ResourcePoolConfigurationService class.

Table 39 – Class: CIM_ResourcePoolConfigurationService

Elements	Requirement	Notes
SystemCreationClassName	Mandatory	None
CreationClassName	Mandatory	None
SystemName	Mandatory	None
Name	Mandatory	None
CreateChildResourcePool	Conditional	See 5.3.1.
AddResourcesToResourcePool	Conditional	See 5.3.2.
RemoveResourcesFromResourcePool	Conditional	See 5.3.4.
DeleteResourcePool	Conditional	See 5.3.2.

5.5.21 CIM_SettingsDefineState

CIM_SettingsDefineState associates an instance of CIM_LogicalDevice that represents a virtual resource and an instance of CIM_ResourceAllocationSettingData that represents the virtualization-specific state of a virtual resource. Table 40 contains the requirements for elements of this class.

Table 40 – Class: CIM_SettingsDefineState

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Reference to an instance of CIM_LogicalDevice that represents a virtual resource Cardinality 0..1
SettingData	Mandatory	**Key:** Reference to an instance of CIM_ResourceAllocationSettingData that represents the virtualization-specific state of a virtual resource Cardinality 0..1

5.5.22 CIM_ServiceAffectsElement

CIM_ServiceAffectsElement associates a CIM_ResourcePool with the service used to manage it. Table 41 defines the requirements for elements of this class.

Table 41 – Class: CIM_ServiceAffectsElement

Elements	Requirement	Notes
AffectedElement	Mandatory	Shall be a reference to an instance of CIM_ResourcePool Cardinality *
AffectingElement	Mandatory	Shall be a reference to an instance of CIM_ResourcePoolConfigurationService Cardinality 1

5.5.23 CIM_SystemDevice

CIM_SystemDevice associates a resource with the system to which it belongs. Table 42 defines the requirements for elements of this class.

Table 42 – Class: CIM_SystemDevice

Elements	Requirement	Notes
GroupComponent	Mandatory	Shall be a reference to an instance of CIM_System Cardinality 1
PartComponent	Mandatory	Shall be a reference to an instance of CIM_LogicalDevice that represents a Resource Cardinality *

6 System Virtualization Profile

Profile Name: System Virtualization

Version: 1.0.0

Organization: DMTF

CIM Schema Version: 2.22

Central Class: CIM_System

Scoping Class: CIM_System

The System Virtualization Profile is an autonomous profile that defines the minimum object model for the representation of host systems. It identifies component profiles that address the allocation of resources. It extends the object model for the representation of virtual systems and virtual resources defined in clause 12.

The central instance and the scoping instance of the System Virtualization Profile shall be an instance of the CIM_System class that represents a host system.

Table 43 lists DMTF management profiles that the profile described in this clause depends on, or that may be used in the context of the profile described in this clause.

Table 43 – Related profiles for the System Virtualization Profile

Profile name	Organization	Version	Relationship	Description
Profile Registration	DMTF	1.0	Mandatory	The DMTF management profile that describes the registration of DMTF management profiles; see 6.2.2.
Virtual System	DMTF	1.0	Mandatory	The autonomous DMTF management profile (see clause 12) that specifies the minimum object model needed for the inspection and basic manipulation of a virtual system; see 6.2.3.
Processor Resource Virtualization	DMTF	1.0	Conditional	The component DMTF management profile (see clause 8) that specifies the allocation of processor resources; see 6.2.2.2.
Memory Resource Virtualization	DMTF	1.0	Conditional	The component DMTF management profile (see clause 9) that specifies the allocation of memory resources; see 6.2.2.2.
Generic Device Resource Virtualization	DMTF	1.0	Conditional	The component DMTF management profile (see clause 13) that specifies the allocation of generic resources; see 6.2.2.2.

6.1 Description

This subclause contains informative text only.

The profile described in this clause defines a top-level object model for the inspection and control of system virtualization facilities provided by host systems. It supports the following range of functions:

- the detection of host systems that provide system virtualization facilities

- the discovery of scoped host resources
- the discovery of scoped resource pools
- the inspection of host system capabilities for
  - the creation and manipulation of virtual systems
  - the allocation of resources of various types
- the inspection of resource pool capabilities
- the discovery of hosted virtual systems
- the inspection of relationships between host entities (host systems, host resources, and resource pools) and virtual entities (virtual systems and virtual resources)
- the creation and manipulation of virtual systems using input configurations, predefined configurations available at the host system, or both
- the creation and manipulation of snapshots that capture the configuration and state of a virtual system at a particular point in time

6.1.1 Profile relationships

A client that is exploiting system virtualization facilities specified by the profile described in this clause needs to be virtualization aware. The specified model keeps that knowledge at an abstract level that is independent of a particular system virtualization platform implementation or technology.

The profile described in this clause complements the Virtual System Profile described in clause 12.

- The profile described in this clause focuses on virtualization aspects related to host systems and their resources, such as modeling the relationships between host resources and virtual resources. Further it addresses virtualization-specific tasks such as the creation or modification of virtual systems and their configurations.
- The Virtual System Profile described in clause 12 defines a top-level object model for the inspection and basic operation of virtual systems. It is a specialization of DSP1052 that defines a management interface for general-purpose computer systems. Consequently, the interface specified for the basic inspection and operation of virtual systems is conformant with that specified for real systems. A client that is exploiting capabilities specified by DSP1052 with respect to virtual systems that are instrument conformant with the Virtual System Profile (described in clause 12) can inherently handle virtual systems like real systems without being virtualization aware.

Figure 7 shows the structure of DMTF management profiles related to system virtualization.

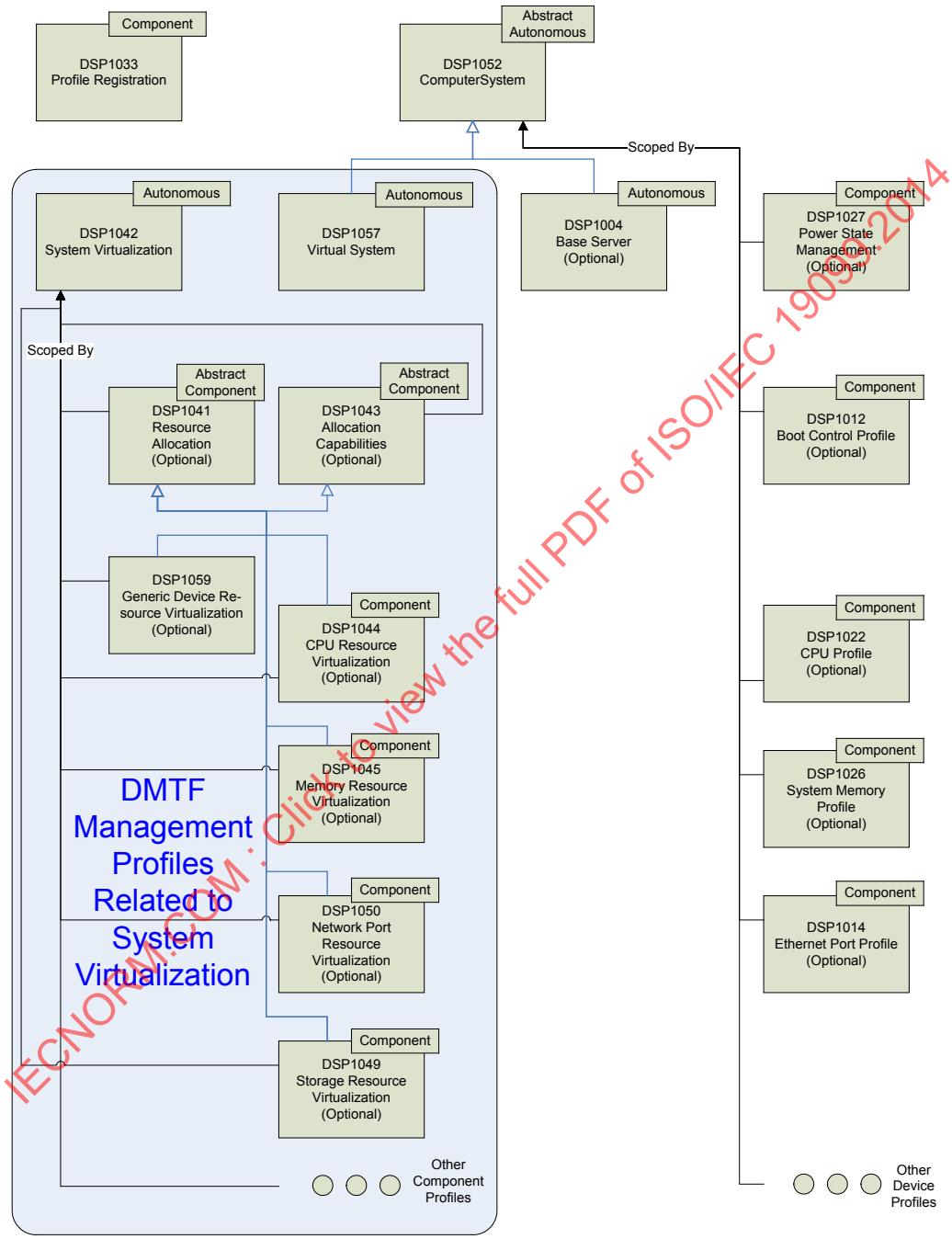

Figure 7 – Profiles related to system virtualization

For example, an implementation that instruments a virtualization platform may implement some of the following DMTF management profiles:

- The System Virtualization Profile

  This profile enables the inspection of host systems, their resources, their capabilities, and their services for creation and manipulation of virtual systems.

- The Virtual System Profile described in clause 12

  The Virtual System Profile enables the inspection of and basic operations on virtual systems.

- Resource-type-specific profiles

  Resource-type-specific profiles enable the inspection and operation of resources for one particular resource type. They apply to both virtual and host resources; they do not cover virtualization-specific aspects of resources. A client may exploit resource-type-specific profiles for the inspection and manipulation of virtual and host resources in a similar manner.

- Resource allocation profiles

  Resource allocation profiles enable the inspection and management of resource allocation requests, allocated resources, and resources available for allocation. Resource allocation profiles are based on the Resource Allocation Profile described in clause 5 and on Allocation Capabilities Profile described in clause 7. Resource allocation profiles are scoped by the profile described in this clause. A client may exploit resource allocation profiles for the inspection of

  – allocated resources

  – allocation dependencies that virtual resources have on host resources and resource pools

  – capabilities that describe possible values for allocation requests

  – capabilities that describe the mutability of resource allocations

  For some resource types, specific resource allocation profiles are specified that address resource-type-specific resource allocation aspects and capabilities. Examples are the Processor Resource Virtualization Profile described in clause 8 and the Storage Resource Virtualization Profile described in clause 10.

  The management of the allocation of basic virtual resources that are not covered by a resource-type-specific resource allocation profile is specified in the Generic Device Resource Virtualization Profile described in clause 13.

6.1.2 System virtualization class schema

Figure 8 shows the complete class schema of the profile described in this clause. It outlines elements that are specified or specialized by the profile described in this clause, as well as the dependency relationships between elements of the profile described in this clause and other profiles. For simplicity in diagrams, the prefix CIM_ has been removed from class and association names.

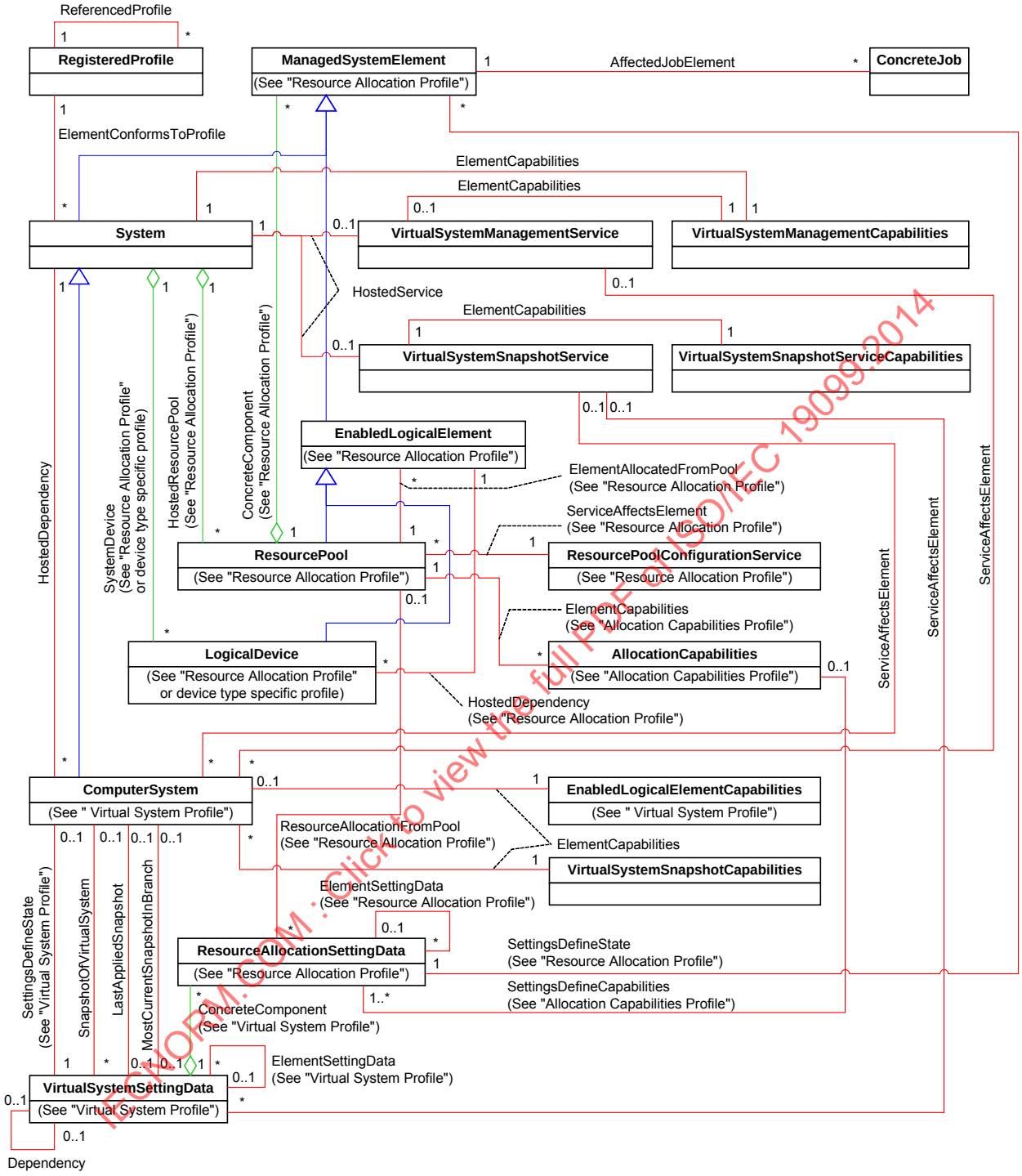

Figure 8 – System Virtualization Profile: Class diagram

The Resource Allocation Profile specifies the use of the following classes and associations:

- the CIM_RegisteredProfile class and the CIM_ElementConformsToProfile association for the advertisement of conformance to this profile
- the CIM_ReferencedProfile association for the representation of a scoping relationship between the profile described in this clause and scoped DMTF management profiles
- the CIM_System class for the representation of host systems
- the CIM_HostedDependency association for the representation of the hosting relationship between a host system and hosted virtual systems
- the CIM_VirtualSystemManagementService class for the representation of virtual system management services available at a host system, providing operations like the creation and modification of virtual systems and their components
- the CIM_HostedService association for the representation of the relationship between a host system and services that it provides
- the CIM_VirtualSystemManagementCapabilities class for the representation of optional features, properties, and methods available for the management of virtual systems hosted by a host system
- the CIM_ElementCapabilities association for the representation of the relationship between a host system, a virtual system or a service, and their respective capabilities
- the CIM_ServiceAffectsElement association for the representation of the relationship between defined services and affected elements like virtual systems or virtual system snapshots
- the CIM_VirtualSystemSettingData class for the representation of snapshots (in addition to the use of that class for the representation of virtual aspects of a virtual system as specified by the Virtual System Profile described in clause 12)
- the CIM_VirtualSystemSnapshotService class for the representation of snapshot-related services available at a host system
- the CIM_VirtualSystemSnapshotServiceCapabilities class for the representation of optional features, properties, and methods available for the management of snapshots of virtual systems
- the CIM_VirtualSystemSnapshotCapabilities class for the representation of optional features, properties, and methods available for the management of snapshots relating to one particular virtual system
- the CIM_SnapshotOfVirtualSystem association for the representation of the relationship between a snapshot of a virtual system and the virtual system itself
- the CIM_Dependency association for dependencies among virtual system snapshots
- the CIM_LastAppliedSnapshot association for the representation of the relationship between a virtual system and the snapshot that was most recently applied to it
- the CIM_MostCurrentSnapshotInBranch association for the representation of the relationship between a virtual system and the snapshot that is the most current snapshot in a sequence of snapshots captured from the virtual system
- the CIM_ConcreteJob class and the CIM_AffectedJobElement association to model a mechanism that allows tracking of asynchronous tasks resulting from operations such as the optional CreateSystem( ) method of the CIM_VirtualSystemManagementService class

In general, any mention of a class in this document means the class itself or its subclasses. For example, a statement such as "an instance of the CIM_LogicalDevice class" implies an instance of the CIM_LogicalDevice class or a subclass of the CIM_LogicalDevice class.

6.1.3 Virtual system configurations

The profile described in this clause extends the use of virtual system configurations. The Virtual System Profile described in clause 12 defines a virtual system configuration as one top-level instance of the CIM_VirtualSystemSettingData class that aggregates zero or more instances of the CIM_ResourceAllocationSettingData class through the CIM_VirtualSystemSettingDataComponent association.

The Virtual System Profile described in clause 12 defines the concept of virtual system configurations and applies it to the following types of virtual system configurations:

- the "State" virtual system configuration, which represents a virtualization-specific state that extends a virtual system representation
- the "Defined" virtual system configuration, which represents virtual system definitions
- the "Next" virtual system configuration, which represents the virtual system configuration that will be used for the next activation of a virtual system

The profile described in this clause applies the concept of virtual system configurations and defines the following additional types of virtual system configurations:

- the "Input" virtual system configuration, which represents configuration information for new virtual systems
- the "Reference" virtual system configuration, which represents configuration information that complements an "Input" virtual system configuration for a new virtual system
- the "Snapshot" virtual system configuration, which represents snapshots of virtual systems

6.1.4 Resource allocation

An allocated resource is a resource subset or resource share that is allocated from a resource pool. An allocated resource is obtained based on a resource allocation request. Both allocated resources and resource allocation requests are represented through instances of the CIM_ResourceAllocationSettingData class.

A virtual resource or a comprehensive set of virtual resources is the representation of an allocated resource. For example, a set of virtual processors represent an allocated processor resource.

Resource allocation is the process of obtaining an allocated resource based on a resource allocation request. The profile described in this clause distinguishes two types of resource allocation:

- Persistent Resource Allocation

  Persistent resource allocation occurs while virtual resources are defined and supporting resources are persistently allocated from a resource pool.

- Transient Resource Allocation

  Transient resource allocation occurs as virtual resources are instantiated and supporting resources are temporarily allocated from a resource pool for the lifetime of the virtual resource instance.

  EXAMPLE 1: Persistent Resource Allocation: File-based virtual disk

  A host file is persistently allocated as the virtual disk is defined. The file remains persistently allocated while the virtual disk remains defined even while the virtual system is not instantiated.

  EXAMPLE 2: Transient Resource Allocation: Host memory

  A contiguous chunk of host memory is temporarily allocated to support virtual memory as the scoping virtual system is instantiated. The memory chunk remains allocated for the time that the virtual system remains instantiated.

EXAMPLE 3: Transient Resource Allocation: I/O bandwidth

An I/O bandwidth is temporarily allocated as the scoping virtual system is instantiated. The I/O bandwidth remains allocated only while the virtual system remains instantiated.

It is a normal situation that within one implementation large numbers of virtual systems are defined such that obtaining the sum of all resource allocation requests would overcommit the implementation's capabilities. Nevertheless, the implementation is able support virtual systems or resources in performing their tasks if it ensures that only a subset of such virtual systems or resources is active at a time that the sum of their allocated resources remains within the implementation's capabilities.

6.1.5 Snapshots

A snapshot is a reproduction of the virtual system as it was at a particular point in the past. A snapshot contains configuration information and may contain state information of the virtual system and its resources, such as the content of virtual memory or the content of virtual disks. A snapshot can be applied back into the virtual system any time, reproducing a situation that existed when the snapshot was captured.

The extent of snapshot support may vary: an implementation may support full snapshots, snapshots that capture the virtual system's disks only, or both. Further, an implementation may impose restrictions on the virtual system state of the source virtual system—for example, supporting the capturing of snapshots only while the virtual system is in the "Defined" state. The extent of snapshot support is modeled through specific capabilities classes.

Implementations may establish relationships between snapshots. For example, snapshots may be ordered by their creation time.

The profile described in this clause specifies mechanisms for the creation, application, and destruction of snapshots. It specifies a snapshot model that enables the inspection of snapshot-related configuration information such as the virtual system configurations that were effective when the snapshot was captured. Relationships between snapshots are also modeled.

The profile described in this clause specifies mechanisms that enable the inspection of configuration information of snapshots and their related virtual systems only. The profile described in this clause does not specify mechanisms for the inspection of the content that was captured in a snapshot, such as raw virtual memory images or raw virtual disk images.

6.2 Implementation

This subclause details the requirements related to classes and their properties for implementations of the profile described in this clause. The CIM Schema descriptions for any referenced element and its subelements apply.

The list of all required methods can be found in 6.3 ("Methods") and the list of all required properties can be found in 6.5 ("CIM elements").

Where reference is made to CIM Schema properties that enumerate values, the numeric value is normative and the descriptive text following it in parentheses is informational. For example, in the statement "If an instance of the CIM_VirtualSystemManagementCapabilities class contains the value 3 (DestroySystemSupported) in an element of the SynchronousMethodsSupported[ ] array property," the value "3" is normative text and "(DestroySystemSupported)" is informational text.

6.2.1 Host system

The CIM_System class shall be used for the representation of host systems. There shall be one instance of the CIM_System class for each host system that is managed conformant to the profile described in this clause.

6.2.2 Profile registration

DSP1033 describes how an implementation of a profile shall advertise that a profile is implemented.

6.2.2.1 The System Virtualization Profile

The implementation of the profile described in this clause shall be indicated by an instance of the CIM_RegisteredProfile class in the CIM Interop namespace. Each instance of the CIM_System class that represents a host system that is manageable through the profile described in this clause shall be a central instance of the profile described in this clause by associating it with the instance of the CIM_RegisteredProfile class through an instance of the CIM_ElementConformsToProfile association.

6.2.2.2 Scoped resource allocation profiles

An implementation of the profile described in this clause may indicate that it is capable of representing the allocation of resources to support virtual resources by implementing scoped resource-allocation DMTF management profiles.

The support of scoped resource-allocation profiles is conditional with respect to the presence of an instance of the CIM_RegisteredProfile class in the Interop namespace that represents the scoped resource–allocation profile implementation and is associated with the instance of the CIM_RegisteredProfile class that represents an implementation of the profile described in this clause through an instance of the CIM_ReferencedProfile association.

Resource-allocation DMTF management profiles are based on the Resource Allocation Profile (see clause 5) and the Allocation Capabilities Profile (see clause 7). The resource-allocation DMTF management profiles that are scoped by the profile described in this clause are listed in Table 43, starting with the Processor Resource Virtualization Profile described in clause 8.

An implementation that provides conditional support for inspecting and managing the allocation of resources of one particular resource type shall apply one of the following implementation approaches:

- If a resource-type-specific resource-allocation DMTF management profile is specified for that resource type, that profile should be implemented.
- If no resource-type-specific resource-allocation DMTF management profile exists at version 1.0 or later, the Generic Device Resource Virtualization Profile described in clause 13 should be implemented.

For any implementation of a scoped-resource-allocation DMTF management profile, all of the following conditions shall be met:

- The instance of the CIM_RegisteredProfile class that represents the implementation of the profile described in this clause and the instance of the CIM_RegisteredProfile class that represents the implementation of the scoped resource-allocation DMTF management profile shall be associated through an instance of the CIM_ReferencedProfile association.
- One of the following conditions regarding profile implementation advertisement shall be met:
  - Central Class Profile Implementation Advertisement:
    Instances of the CIM_ElementConformsToProfile association shall associate each instance of the CIM_ResourcePool class that is a central instance of the scoped-resource-allocation DMTF management profile with the instance of the CIM_RegisteredProfile class that represents an implementation of the scoped-resource-allocation DMTF management profile.
  - Scoping Class Profile Implementation Advertisement:
    No instances of the CIM_ElementConformsToProfile association shall associate any instance of the CIM_ResourcePool class that is a central instance of the scoped-resource-allocation DMTF management profile with the instance of the CIM_RegisteredProfile class

that represents an implementation of the scoped-resource-allocation DMTF management profile.

6.2.3 Representation of hosted virtual systems

The profile described in this clause strengthens the requirements for the representation of virtual system configurations specified by the Virtual System Profile described in clause 12 for hosted virtual systems.

6.2.3.1 Profile conformance for hosted virtual systems

Any virtual system that is hosted by a conformant host system shall be represented by an instance of the CIM_ComputerSystem class that is a central instance of the Virtual System Profile described in clause 12. That instance shall be associated with the instance of the CIM_System class that represents the conformant host system through an instance of the CIM_HostedDependency association.

6.2.3.2 CIM_VirtualSystemSettingData.VirtualSystemType property

The value of the VirtualSystemType property shall be equal to an element of the VirtualSystemTypesSupported[ ] array property in the instance of the CIM_VirtualSystemManagementCapabilities class that is associated with the instance of the CIM_VirtualSystemManagementService class that represents the host system, or shall be NULL if the value of the VirtualSystemTypesSupported[ ] array property is NULL (see 6.2.4.2).

6.2.4 Virtual system management capabilities

This subclause models capabilities of virtual system management in terms of the CIM_VirtualSystemManagementCapabilities class.

6.2.4.1 CIM_VirtualSystemManagementCapabilities class

An instance of the CIM_VirtualSystemManagementCapabilities class shall be used to represent the virtual system management capabilities of a host system. That instance shall be associated with the instance of the CIM_System class that represents the host system through the CIM_ElementCapabilities association.

6.2.4.2 CIM_VirtualSystemManagementCapabilities.VirtualSystemTypesSupported[ ] array property

The implementation of the VirtualSystemTypesSupported[ ] array property is optional. The VirtualSystemTypesSupported[ ] array property should be implemented.

If the VirtualSystemTypesSupported[ ] array property is implemented, the provisions in this subclause apply.

Array values shall designate the set of supported virtual system types. If the VirtualSystemTypesSupported[ ] array property is not implemented (has a value of NULL), the implementation does not externalize the set of implemented virtual system types, but internally still may exhibit different types of virtual systems.

6.2.4.3 CIM_VirtualSystemManagementCapabilities.SynchronousMethodsSupported[ ] array property

The implementation of the SynchronousMethodsSupported[ ] array property is optional. The SynchronousMethodsSupported[ ] array property should be implemented.

If the SynchronousMethodsSupported[ ] array property is implemented, the provisions in this subclause apply.

Array values shall designate the set of methods of the CIM_VirtualSystemManagementService class that are implemented with synchronous behavior only. A NULL value or an empty value set shall be used to indicate that no methods are implemented with synchronous behavior. If a method is designated within the value set of the SynchronousMethodsSupported[ ] property, that method shall always exhibit synchronous behavior and shall not be designated within the value set of the AsynchronousMethodsSupported[ ] property.

6.2.4.4 CIM_VirtualSystemManagementCapabilities.AsynchronousMethodsSupported[ ] array property

The implementation of the AsynchronousMethodsSupported[ ] array property is optional. The AsynchronousMethodsSupported[ ] array property should be implemented.

If the AsynchronousMethodsSupported[ ] array property is implemented, the provisions in this subclause apply.

Array values shall designate the set of methods of the CIM_VirtualSystemManagementService class that are implemented with synchronous and potentially with asynchronous behavior. A NULL value or an empty value set shall be used to indicate that no methods are implemented with asynchronous behavior. If a method is designated with a value in the AsynchronousMethodsSupported[ ] array property, it may show either synchronous or asynchronous behavior.

6.2.4.5 CIM_VirtualSystemManagementCapabilities.IndicationsSupported[ ] array property

The implementation of the IndicationsSupported[ ] array property is optional. The IndicationsSupported[ ] array property should be implemented.

If the IndicationsSupported[ ] array property is implemented, the provisions in this subclause apply.

Array values shall designate the set of types of indications that are implemented. A NULL value or an empty value set shall be used to indicate that indications are not implemented.

6.2.4.6 Grouping Rules for implementations of methods of the CIM_VirtualSystemManagementService class

The grouping rules specified in this subclause shall be applied for implementations of methods of the CIM_VirtualSystemManagementService class. Within a group either all methods or no method at all shall be implemented; nevertheless synchronous and asynchronous behavior may be mixed.

6.2.4.6.1 Virtual system definition and destruction

If virtual system definition and destruction are implemented, the DefineSystem( ) and DestroySystem( ) methods of the CIM_VirtualSystemManagementService class shall be implemented, and the values 2 (DefineSystemSupported) and 3 (DestroySystemSupported) shall be set in the SynchronousMethodsSupported[ ] or AsynchronousMethodsSupported[ ] array properties within the instance of the CIM_VirtualSystemManagementCapabilities class that describes capabilities of the implementation.

If virtual system definition and destruction are not implemented, the values 2 (DefineSystemSupported) and 3 (DestroySystemSupported) shall not be set in the SynchronousMethodsSupported[ ] or AsynchronousMethodsSupported[ ] array properties of the instance of the CIM_VirtualSystemManagementCapabilities class that describes the virtual system management capabilities of the host system.

6.2.4.6.2 Virtual resource addition and removal

If the addition and removal of virtual resources to or from virtual systems are implemented, the AddResourceSettings( ) and RemoveResourceSettings( ) methods of the CIM_VirtualSystemManagementService class shall be implemented, and the values 1 (AddResourceSettingsSupported) and 7 (RemoveResourceSettingsSupported) shall be set in the SynchronousMethodsSupported[ ] or AsynchronousMethodsSupported[ ] array properties of the instance of the CIM_VirtualSystemManagementCapabilities class that describes the virtual system management capabilities of the host system.

If the addition and removal of virtual resources to virtual systems is not implemented, the values 1 (AddResourceSettingsSupported) and 7 (RemoveResourceSettingsSupported) shall not be set in the SynchronousMethodsSupported[ ] or AsynchronousMethodsSupported[ ] array properties of the instance of the CIM_VirtualSystemManagementCapabilities class that describes the virtual system management capabilities of the host system.

6.2.4.6.3 Virtual system and resource modification

If the modification of virtual systems and virtual resources is implemented, the ModifyResourceSettings( ) and ModifySystemSettings( ) methods of the CIM_VirtualSystemManagementService class shall be implemented, and the values 5 (ModifyResourceSettingsSupported) and 6 (ModifySystemSettingsSupported) shall be set in the SynchronousMethodsSupported[ ] or AsynchronousMethodsSupported[ ] array properties of the instance of the CIM_VirtualSystemManagementCapabilities class that describes the virtual system management capabilities of the host system.

If the modification of virtual systems and virtual resources is not implemented, the values 5 (ModifyResourceSettingsSupported) and 6 (ModifySystemSettingsSupported) shall not be set in the SynchronousMethodsSupported[ ] or AsynchronousMethodsSupported[ ] array properties of the instance of the CIM_VirtualSystemManagementCapabilities class that describes the virtual system management capabilities of the host system.

6.2.5 Virtual system definition and modification

The profile described in this clause specifies methods for the definition and modification of virtual systems. These method specifications use the CIM_VirtualSystemSettingData class for the parameterization of system-specific properties. Subsequent subclauses specify:

- how a client shall prepare instances of the CIM_VirtualSystemSettingData class that are used as a parameter for a method that defines or modifies a virtual system
- how an implementation shall interpret instances of the CIM_VirtualSystemSettingData class that are used as a parameter for a method that defines or modifies a virtual system

Definition requests for virtual systems are modeled through the CIM_VirtualSystemManagementService.DefineSystem( ) method, and modification requests for virtual system properties are modeled through the CIM_VirtualSystemManagementService.ModifySystemSettings( ) method.

6.2.5.1 CIM_VirtualSystemSettingData.InstanceID property

A client shall set the value of the InstanceID property to NULL if the instance of the CIM_VirtualSystemSettingData class is created locally. A client shall not modify the value of the InstanceID property in an instance of the CIM_VirtualSystemSettingData class that was received from an implementation and is sent back to the implementation as a parameter of a modification method.

The structure of the value of the InstanceID property is implementation specific. A client shall treat the value as an opaque entity and shall not depend on the internal structure of the value.

An implementation shall use a non-NULL value to identify an existing instance of the CIM_VirtualSystemSettingData class. If the value does not identify an instance of the CIM_VirtualSystemSettingData class, an implementation shall return a return code that indicates an invalid parameter (see 6.3.2.4.3).

6.2.5.2 CIM_VirtualSystemSettingData.ElementName property

The implementation of the ElementName property is optional.

If the ElementName property is implemented for virtual system definition and modification, the provisions in this subclause apply.

A client may set the value of the ElementName property to assign a user-friendly name to a virtual system.

In definition and modification requests, an implementation shall use the value of the ElementName property to assign a user-friendly name to the new virtual system. The user-friendly name does not have to be unique within the set of virtual systems that are defined at the host system.

If the implementation supports modification requests that affect the value of the ElementName property, the implementation shall support the CIM_EnabledLogicalElementCapabilities class for virtual systems as specified in DSP1052.

6.2.5.3 CIM_VirtualSystemSettingData.VirtualSystemIdentifier property

The implementation of the VirtualSystemIdentifier property is optional.

If the VirtualSystemIdentifier property is implemented for virtual system definition and modification, the provisions in this subclause apply.

A client should set the value of the VirtualSystemIdentifier property to explicitly request an identifier for the new virtual system. A client may set the value of the VirtualSystemIdentifier property to NULL.

An implementation shall use the value of the VirtualSystemIdentifier property to assign an identifier to the new virtual system. If the value of the VirtualSystemIdentifier property is NULL, the value of the VirtualSystemIdentifier property for the new virtual system is unspecified (implementation dependent).

Some implementations may accept an implementation-dependent pattern that controls the assignment of a value to the VirtualSystemIdentifier property. For example, an implementation might interpret a regular expression like "^VM\d{1,6}\$" to assign a value to the VirtualSystemIdentifier property that starts with the letters "VM" and is followed by at least one and not more than six digits.

6.2.5.4 CIM_VirtualSystemSettingData.VirtualSystemType property

The implementation of the VirtualSystemType property is optional.

If the VirtualSystemType property is implemented for virtual system definition and modification, the provisions in this subclause apply.

A client may set the value of the VirtualSystemType property to explicitly request a virtual system type for the new virtual system. A client may set the value of the VirtualSystemType property to NULL, requesting the implementation to assign a virtual system type according to rules specified in this subclause. If requesting a value other than NULL, the client should determine the list of valid system types in advance (see 6.4.2.7).

An implementation shall use the value of the VirtualSystemType property to assign a type to the new virtual system. If the value of the VirtualSystemType property is NULL, the implementation shall assign a virtual system type in an implementation-dependent way. If the requested virtual system type is not sup-

ported, an implementation shall fail the method execution with an error code of 4 (Method execution failed because invalid parameters were specified by the client).

6.2.6 Virtual resource definition and modification

The profile described in this clause specifies how to define and modify virtual resources using methods of the virtual system management service. In these method specifications, the CIM_ResourceAllocationSettingData class is used for parameterization of resource allocation specific properties. For specifications that define the use of the CIM_ResourceAllocationSettingData class, see the Resource Allocation Profile (clause 5), the Allocation Capabilities Profile (clause 7), and profiles that specialize these (for example, the Generic Device Resource Virtualization Profile described in clause 13). The Resource Allocation Profile (clause 5) describes the use of the CIM_ResourceAllocationSettingData class, and the Allocation Capabilities Profile (clause 7) introduces the concept of allowing a client to determine the acceptable value sets for values of properties of the CIM_ResourceAllocationSettingData class in virtual resource definition and modification requests.

6.2.7 Virtual system snapshots

This subclause models the representation and manipulation of snapshots of virtual systems.

The implementation of virtual system snapshots is optional.

If virtual system snapshots are implemented, the provisions in this subclause apply.

6.2.7.1.1 Virtual system snapshot service and capabilities

This subclause models elements of virtual system snapshot management in terms of the CIM_VirtualSystemSnapshotService class and the CIM_VirtualSystemSnapshotServiceCapabilities class.

6.2.7.1.2 Virtual system snapshots

The implementation of virtual system snapshots is optional.

If virtual system snapshots are implemented, the provisions in this subclause apply.

The implementation includes the creation, destruction, and application of virtual system snapshots.

If virtual system snapshots are implemented, the following conditions shall be met:

- the CIM_VirtualSystemSnapshotService class shall be implemented and the following methods shall be implemented:
  - CreateSnapshot( ), for at least one type of snapshot
  - DestroySnapshot( )
  - ApplySnapshot( )
- There shall be exactly one instance of the CIM_VirtualSystemSnapshotService class associated to the central instance of the profile described in this clause through an instance of the CIM_HostedService association.

If virtual system snapshots are not implemented, the CIM_VirtualSystemSnapshotService class shall not be implemented.

6.2.7.1.3 CIM_VirtualSystemSnapshotServiceCapabilities class

The provisions in this subclause are conditional.

Condition: Virtual system snapshots are implemented; see 6.2.7.1.2.

If the CIM_VirtualSystemSnapshotServiceCapabilities class is implemented, the provisions in this subclause apply.

An instance of the CIM_VirtualSystemSnapshotServiceCapabilities class shall be used to represent the capabilities of the virtual system snapshot service of a host system. The instance shall be associated with the instance of the CIM_VirtualSystemSnapshotService class that represents the virtual system snapshot service through the CIM_ElementCapabilities association.

In the instance of the CIM_VirtualSystemSnapshotServiceCapabilities class that describes virtual system snapshot service, all of the following values shall be set in either the SynchronousMethodsSupported[ ] array property or the AsynchronousMethodsSupported[ ] array property:

- 2 (CreateSnapshotSupported)
- 3 (DestroySnapshotSupported)
- 4 (ApplySnapshotSupported)

The implementation of the SynchronousMethodsSupported[ ] array property is conditional with respect to at least one of the snapshot methods being implemented with synchronous behavior. A NULL value or an empty value set shall be used to indicate that no methods are implemented with synchronous behavior. If a method is designated within the value set of the SynchronousMethodsSupported[ ] property, that method shall always exhibit synchronous behavior and shall not be designated within the value set of the AsynchronousMethodsSupported[ ] property.

The implementation of the AsynchronousMethodsSupported[ ] array property is conditional with respect to at least one of the snapshot methods being implemented with asynchronous behavior. A NULL value or an empty value set shall be used to indicate that no methods are implemented with asynchronous behavior.

Further the SnapshotTypesSupported[ ] array property shall have a non-NULL value and contain at least one element. Each element of the SnapshotTypesSupported[ ] array property shall designate one supported type of snapshot.

6.2.7.2 Virtual system snapshot representation

The provisions in this subclause are conditional.

Condition: Virtual system snapshots are implemented; see 6.2.7.1.2.

If the representation of virtual system snapshots is implemented, the provisions in this subclause apply.

Snapshots of virtual systems shall be represented by instances of the CIM_VirtualSystemSettingData class. Each such instance shall be associated with the instance of the CIM_ComputerSystem class that represents the virtual system that was the source of the snapshot through an instance of the CIM_SnapshotOfVirtualSystem association.

6.2.7.3 Designation of the last applied snapshot

The provisions in this subclause are conditional.

Condition: Virtual system snapshots are implemented; see 6.2.7.1.2.

If the designation of the last applied snapshot is implemented, the provisions in this subclause apply.

If a snapshot was applied to a virtual system, an instance of the CIM_LastAppliedSnapshot association shall connect the instance of the CIM_ComputerSystem class that represents the virtual system and the instance of the CIM_VirtualSystemSettingData class that represents the snapshot. The association instance shall be actualized as different snapshots are applied.

6.2.7.4 Designation of the most current snapshot in branch

The implementation of the representation the most current snapshot in a branch is conditional.

Condition: Virtual system snapshots are implemented; see 6.2.7.1.2.

If the designation of the most current snapshot in a branch is implemented, the provisions in this subclause apply.

A branch of snapshots taken from a virtual system is started in one of two ways:

- A virtual system snapshot is applied to a virtual system.

  In this case, the virtual system snapshot becomes the most current snapshot of a newly started branch.

- A virtual system snapshot is captured from a virtual system.

  In this case, the virtual system snapshot becomes the most current snapshot in the branch. If no branch exists, a new branch is created.

6.2.7.5 Virtual system snapshot capabilities

The provisions in this subclause are optional.

If virtual system snapshot capabilities are implemented, the provisions in this subclause apply.

This subclause models snapshot related capabilities of a virtual system in terms of the CIM_VirtualSystemSnapshotCapabilities class.

6.2.7.5.1 CIM_VirtualSystemSnapshotCapabilities.SnapshotTypesEnabled[ ] array property

An implementation shall use the SnapshotTypesEnabled[ ] array property to convey information about the enablement of snapshot types The value set of the SnapshotTypesEnabled[ ] array property shall designate those snapshot types that are presently enabled (that is, may be invoked by a client).

NOTE Elements may be added and removed from the array property as respective snapshot types are enabled for the virtual system; the conditions for such changes are implementation specific.

6.2.7.5.2 CIM_VirtualSystemSnapshotCapabilities.GuestOSNotificationEnabled property

The implementation of the GuestOSNotificationEnabled property is optional.

If the GuestOSNotificationEnabled property is implemented, the provisions in this subclause apply.

An implementation may use the GuestOSNotificationEnabled property to convey information about the capability of the guest operating system that is running within a virtual system to receive notifications about an imminent snapshot operation. The behavior of the guest operating system in response to such a notification is implementation dependent. For example, the guest operating system may temporarily suspend operations on virtual resources that might interfere with the snapshot operation.

6.3 Methods

This subclause defines extrinsic methods and profile conventions for intrinsic methods. The specifications provided in this subclause apply in addition to the descriptions provided in the CIM Schema.

6.3.1 General behavior of extrinsic methods

This subclause models behavior applicable to all extrinsic methods that are specified in the profile described in this clause.

6.3.1.1 Resource allocation requests

Some methods specify the ResourceSettings[ ] array parameter. If set to a value other than NULL, each element of the ResourceSettings[ ] array parameter shall contain an embedded instance of the CIM_ResourceAllocationSettingData class that describes a resource allocation request for a virtual resource or coherent set of virtual resources.

The use of the CIM_ResourceAllocationSettingData class as input for operations is specified in the Resource Allocation Profile (see clause 5).

One instance of the CIM_ResourceAllocationSettingData class may affect one virtual resource or a coherent set of virtual resources. For example, one instance of CIM_ResourceAllocationSettingData that has the value of the ResourceType property set to 3 (Processor) and the value of the VirtualQuantity property set to 2 requests the allocation of two virtual processors.

If one or more resources are not available, or not completely available, during the execution of a method that requests the allocation of persistently allocated resources into a virtual system configuration, the implementation may deviate from requested values, may ignore virtual resource allocation requests, or both as long as the resulting virtual system is or remains potentially operational. Otherwise, the implementation shall fail the method execution.

6.3.1.2 Method results

If a particular method is not implemented, a value of 1 (Not Supported) shall be returned.

If synchronous execution of a method succeeds, the implementation shall set a return value of 0 (Completed with No Error).

If synchronous execution of a method fails, the implementation shall set a return value of 2 (Failed) or a more specific return code as specified with the respective method.

If a method is executed as an asynchronous task, the implementation shall perform all of the following actions:

- Set a return value of 4096 (Job Started).
- Set the value of the Job output parameter to refer to an instance of the CIM_ConcreteJob class that represents the asynchronous task.
- Set the values of the JobState and TimeOfLastStateChange properties in that instance to represent the state and last state change time of the asynchronous task.

In addition, the implementation may present state change indications as task state changes occur.

If the method execution as an asynchronous task succeeds, the implementation shall perform all of the following actions:

- Set the value of the JobState property to 7 (Completed).
- Provide an instance of the CIM_AffectedJobEntity association with property values set as follows:
  - The value of the AffectedElement property shall refer to the object that represents the top-level entity that was created or modified by the asynchronous task. For example, for the DefineSystem( ) method, this is an instance of the CIM_ComputerSystem class, and for the CreateSnapshot( ) method, this is an instance of the CIM_VirtualSystemSettingData class that represents a snapshot of a virtual system.
  - The value of the AffectingElement property shall refer to the instance of the CIM_ConcreteJob class that represents the completed asynchronous task.

– The value of the first element in the ElementEffects[ ] array property (ElementEffects[0]) shall be set to 5 (Create) for the DefineSystem( ) or CreateSnapshot( ) methods. Otherwise, this value shall be 0 (Unknown).

If the method execution as an asynchronous task fails, the implementation shall set the value of the JobState property to 9 (Killed) or 10 (Exception).

6.3.1.3 Asynchronous processing

An implementation may support asynchronous processing of some methods specified in the CIM_VirtualSystemManagementService class.

6.3.1.3.1 General requirements

All of the following conditions shall be met:

- Elements that convey information about which methods of the CIM_VirtualSystemManagementService class are implemented for asynchronous execution within an implementation are modeled in 6.2.4.4.
- Elements that convey information about which methods of the CIM_VirtualSystemSnapshotService class are implemented for asynchronous execution within an implementation are modeled in 6.2.7.1.2.
- Elements that convey information about whether a method is executed asynchronously are modeled in 6.3.1.2.

6.3.1.3.2 Job parameter

The implementation shall set the value of the Job parameter as a result of an asynchronous execution of a method of the CIM_VirtualSystemManagementService as follows:

- If the method execution is performed synchronously, the implementation shall set the value to NULL.
- If the method execution is performed asynchronously, the implementation shall set the value to refer to the instance of the CIM_ConcreteJob class that represents the asynchronous task.

6.3.2 Methods of the CIM_VirtualSystemManagementService class

This subclause models virtual system management services in terms of methods of the CIM_VirtualSystemManagementService class.

6.3.2.1 CIM_VirtualSystemManagementService.DefineSystem( ) method

The implementation of the DefineSystem( ) method is conditional.

Condition: The definition and destruction of virtual systems is implemented; see 6.2.4.6.1.

If the DefineSystem( ) method is implemented, the provisions in this subclause apply; in addition behavior applicable to all extrinsic methods is specified in 6.3.1.2.

The execution of the DefineSystem( ) method shall effect the creation of a new virtual system definition as specified through the values of the SystemSettings parameter, the values of elements in the ResourceSettings[ ] array parameter and elements of the configuration referred to by the value of the ReferencedConfiguration parameter, and through default values that are established within the implementation.

Table 44 contains requirements for parameters of this method.

Table 44 – DefineSystem( ) method: Parameters

Qualifiers	Name	Type	Description/Values
IN	SystemSettings	string	See 6.3.2.1.2.
IN	ResourceSettings[ ]	string	See 6.3.2.1.3.
IN	ReferencedConfiguration	CIM_VirtualSystemSettingData REF	See 6.3.2.1.4.
OUT	ResultingSystem	CIM_ComputerSystem REF	See 6.3.2.1.5.
OUT	Job	CIM_ConcreteJob REF	See 6.3.1.3.2.

6.3.2.1.1 Value preference rules

The DefineSystem( ) method facilitates the definition of a new virtual system at the host system, based on client requirements specified through one or more virtual system configurations:

- "Input" virtual system configuration

  The "Input" virtual system configuration is prepared locally by the client and provided in the form of embedded instances of the CIM_VirtualSystemSettingData class in the SystemSettings parameter and embedded instances of the CIM_ResourceAllocationSettingData class as values for elements of the ResourceSettings[ ] array parameter.

- "Reference" virtual system configuration

  The "Reference" virtual system configuration is a "Defined" virtual system configuration that already exists within the implementation; it is referenced by the ReferencedConfiguration parameter.

An implementation shall define the virtual system based on "Input" and "Reference" configuration. It may extend a virtual system definition beyond client requirements based on implementation-specific rules and requirements.

If only the "Reference" virtual system configuration is provided by the client, the implementation shall create a copy or cloned configuration of the "Reference" virtual system configuration.

If both configurations are provided by the client, the implementation shall give the "Input" virtual system configuration preference over the "Reference" configuration. An implementation may support this behavior at two levels:

- The basic level supports the addition of resource allocations that were not requested by elements of the ResourceSettings[ ] array parameter, but that are defined in the "Reference" virtual system configuration.
- The advanced level, in addition, supports amending incomplete resource requests.

  In this case the correlation of instances of the CIM_ResourceAllocationSettingData class in the "Input" configuration and in the "Reference" configuration shall be established through the value of the InstanceID parameter. If the value of the InstanceID parameter is identical for an instance in the "Input" configuration and an instance in the "Reference" configuration, these instances together describe one virtual resource allocation request, such that non-NULL property values specified in the "Input" configuration override those specified in the "Reference" configuration.

If no value is specified for a property in the "Input" configuration or in the "Reference" configuration, the implementation may exhibit an implementation-dependent default behavior. The Generic Device

Resource Virtualization Profile described in clause 13 and resource-type-specific resource allocation DMTF management profiles may specify resource-type-specific behavior.

If the DefineSystem( ) method is called without input parameters, the implementation may exploit a default behavior or may fail the method execution.

NOTE A client may inspect the "Reference" virtual system configuration before invoking the DefineSystem( ) method. (See respective use cases for the Virtual System Profile in 12.4.)

6.3.2.1.2 SystemSettings parameter

A client should set the value of the SystemSettings parameter with an embedded instance of the CIM_VirtualSystemSettingData class that describes requested virtual system settings. The client may set the value of the SystemSettings parameter to NULL, requesting the implementation to select input values based on the rules specified in 6.3.2.1.1.

An implementation shall interpret the value of the SystemSettings parameter as the system part of an "Input" virtual system configuration, and apply the rules specified in 6.3.2.1.1.

The use of the CIM_VirtualSystemSettingData class as input for operations specified by the profile described in this clause is specified in 6.5.22.

6.3.2.1.3 ResourceSettings[ ] array parameter

A client should set the ResourceSettings[ ] array parameter and apply the specifications given in 6.3.1.1. The client may set the value of the ResourceSettings[ ] array parameter to NULL or provide an empty array, requesting the implementation to define a default set of virtual resources (see 6.3.2.1.1).

An implementation shall interpret the value of the ResourceSettings[ ] array parameter as the resource part of an "Input" virtual system configuration, and apply the value preference rules specified in 6.3.2.1.1.

6.3.2.1.4 ReferencedConfiguration parameter

A client may set a value of the ReferencedConfiguration parameter to refer to an existing "Defined" virtual system configuration. A client may set the value of the ReferencedConfiguration parameter to NULL, indicating that a "Reference" configuration shall not be used.

An implementation shall use the "Reference" virtual system configuration according to the rules specified in 6.3.2.1.1.

6.3.2.1.5 ResultingSystem parameter

The implementation shall set the value of the ResultingSystem parameter as follows:

- If the method execution is performed synchronously and is successful, the value is set to reference the instance of the CIM_ComputerSystem class that represents the newly defined virtual system.
- If the method execution is performed synchronously and fails, or if the method execution is performed asynchronously, the value is set to NULL.

6.3.2.1.6 Return codes

An implementation shall indicate the result of the method execution by using the return code values specified in Table 45.

Table 45 – DefineSystem( ) method: Return code values

Value	Description
0	Method execution was successful.
1	Method is not supported.
2	Method execution failed.
3	Method execution failed because a timeout condition occurred.
4	Method execution failed because invalid parameters were specified by the client.
4096	Method execution is performed asynchronously. The specifications given in 6.3.1.3 apply.

6.3.2.2 CIM_VirtualSystemManagementService.DestroySystem( ) method

The implementation of the DestroySystem( ) method is conditional.

Condition: The definition and destruction of virtual systems is implemented; see 6.2.4.6.1.

If the DestroySystem( ) method is implemented, the provisions in this subclause apply; in addition behavior applicable to all extrinsic methods is specified in 6.3.1.2.

The execution of the DestroySystem( ) method shall effect the destruction of the referenced virtual system and all related virtual system configurations, including snapshots.

Table 46 contains requirements for parameters of this method.

Table 46 – DestroySystem( ) method: Parameters

Qualifiers	Name	Type	Description/Values
IN	AffectedSystem	CIM_ComputerSystem REF	See 6.3.2.2.1.
OUT	Job	CIM_ConcreteJob REF	See 6.3.1.3.2.

6.3.2.2.1 AffectedSystem parameter

A client shall set a value of the AffectedSystem parameter to refer to the instance of the CIM_ComputerSystem class that represents the virtual system to be destroyed.

An implementation shall interpret the value of the AffectedSystem parameter to identify the virtual system that is to be destroyed.

6.3.2.2.2 Return codes

An implementation shall indicate the result of the method execution by using the return code values specified in Table 47.

Table 47 – DestroySystem( ) method: Return code values

Value	Description
0	Method execution was successful.
1	Method is not supported.
2	Method execution failed.
3	Method execution failed because a timeout condition occurred.
4	Method execution failed because the system could not be found.
5	Method execution failed because the affected system is in a state in which the implementation rejects destruction.
4096	Method execution is performed asynchronously. The specifications given in 6.3.1.3 apply.

6.3.2.3 CIM_VirtualSystemManagementService.AddResourceSettings( ) method (Conditional)

The implementation of the AddResourceSettings( ) method is conditional.

Condition: The addition and the removal of virtual resources to virtual systems is implemented; see 6.2.4.6.2.

If the AddResourceSettings( ) method is implemented, the provisions in this subclause apply; in addition behavior applicable to all extrinsic methods is specified in 6.3.1.2.

The execution of the AddResourceSettings( ) method shall effect the entry of resource allocation requests or resource allocations provided through the ResourceSettings[ ] array parameter in the affected virtual system configuration.

Table 48 contains requirements for parameters of this method.

Table 48 – AddResourceSettings( ) method: Parameters

Qualifiers	Name	Type	Description/Values
IN	AffectedConfiguration	CIM_VirtualSystemSettingData REF	See 6.3.2.3.1.
IN	ResourceSettings[ ]	string	See 6.3.2.3.2.
OUT	ResultingResourceSettings[ ]	CIM_ResourceAllocationSettingData REF	See 6.3.2.3.3.
OUT	Job	CIM_ConcreteJob REF	See 6.3.1.3.2.

6.3.2.3.1 AffectedConfiguration parameter

A client shall set a value of AffectedConfiguration parameter to refer to the instance of the CIM_VirtualSystemSettingData class that represents the virtual system configuration that receives new resource allocations.

An implementation shall interpret the value of the AffectedConfiguration parameter to identify the virtual system configuration that receives new resource allocations.

6.3.2.3.2 ResourceSettings[ ] array parameter

A client shall set the ResourceSettings[ ] parameter containing one or more input instances of the CIM_ResourceAllocationSettingData class as specified in a profile based on the Resource Allocation Profile (see clause 5) and on the Allocation Capabilities Profile (see clause 7), such as for example the Processor Resource Virtualization Profile described in clause 8 or the Storage Resource Virtualization Profile described in clause 10.

If the value of the InstanceID property in any of the input CIM_ResourceAllocationSettingData instances is other than NULL, that value shall be ignored; however, the remaining values of the input instance shall be respected as defined in the resource-type-specific resource allocation profile.

An implementation shall apply the specifications given in 6.3.1.1.

6.3.2.3.3 ResultingResourceSettings[ ] array parameter

The implementation shall set the value of the ResultingResourceSettings[ ] array parameter as follows:

- to an array of references to instances of the CIM_ResourceAllocationSettingData class that represent resource allocations that were obtained during the execution of the method
- to NULL, if the method is executed synchronously and fails, or if the method is executed asynchronously

6.3.2.3.4 Return codes

An implementation shall indicate the result of the method execution by using the return code values specified in Table 49.

Table 49 – AddResourceSettings( ) method: Return code values

Value	Description
0	Method execution was successful.
1	Method is not supported.
2	Method execution failed.
3	Method execution failed because a timeout condition occurred.
4	Method execution failed because invalid parameters were specified by the client.
4096	Method execution is performed asynchronously. The specifications given in 6.3.1.3 apply.

6.3.2.4 CIM_VirtualSystemManagementService.ModifyResourceSettings( ) method

The implementation of the ModifyResourceSettings( ) method is conditional.

Condition: The modification of virtual systems and resources is implemented; see 6.2.4.6.3.

If the ModifyResourceSettings( ) method is implemented, the provisions in this subclause apply; in addition behavior applicable to all extrinsic methods is specified in 6.3.1.2.

If implemented, the execution of the ModifyResourceSettings( ) method shall effect the modification of resource allocation requests that exist, with the implementation using instances of the CIM_ResourceAllocationSettingData class that are passed in through values of elements of the ResourceSettings[ ] array parameter.

The execution of the ModifyResourceSettings( ) method shall effect the modification of resource allocations or resource allocation requests, such that non-key and non-NULL values of instances of the CIM_ResourceAllocationSettingData class provided as values for elements of the ResourceSettings[ ] array parameter override respective values in instances identified through the InstanceID property.

Table 50 contains requirements for parameters of this method.

Table 50 – ModifyResourceSettings( ) method: Parameters

Qualifiers	Name	Type	Description/Values
IN	ResourceSettings[ ]	string	See 6.3.2.4.1.
OUT	ResultingResourceSettings[ ]	CIM_ResourceAllocationSettingData REF	See 6.3.2.4.2.
OUT	Job	CIM_ConcreteJob REF	See 6.3.1.3.2.

6.3.2.4.1 ResourceSettings[ ] parameter

The specifications in 6.3.1.1 apply.

A client shall set the ResourceSettings[ ] parameter. Any instance of the CIM_ResourceAllocationSettingData class that is passed in as a value for elements of the ResourceSettings[ ] array parameter shall conform to all of the following conditions:

- It shall represent requests for the modification of virtual resource state extensions, virtual resource definitions scoped by one particular virtual system, or both.
- It shall have a valid non-NULL value in the InstanceID property that identifies a respective instance of the CIM_ResourceAllocationSettingData class that represents an existing resource allocation or resource allocation request within the implementation. This should be assured through the execution of previously executed retrieve operations, such as the execution of extrinsic methods or intrinsic CIM operations that yield respective instances of the CIM_ResourceAllocationSettingData class. For example, the client may use the intrinsic GetInstance( ) CIM operation.

  The client shall modify such instances locally to reflect the desired modifications and finally pass them back in as elements of the ResourceSettings[ ] array parameter. Modifications shall not be applied to the InstanceID property that is the key property of the CIM_ResourceAllocationSettingData class. Further restriction may apply, such as from resource-type-specific resource allocation DMTF management profiles.

An implementation shall apply the specifications given in 6.3.1.1. The implementation shall ignore any element of the ResourceSettings[ ] array property that does not identify, through the value of the InstanceID key property, an existing instance of the CIM_ResourceAllocationSettingData class within the implementation.

6.3.2.4.2 ResultingResourceSettings[ ] parameter

The implementation shall set the value of the ResultingResourceSettings[ ] array parameter as follows:

- If the method was executed asynchronously, the value shall be set to NULL.
- If the method was executed synchronously and one or more resources were successfully modified, for each successfully modified resource one element in the returned array shall reference the instance of the CIM_ResourceAllocationSettingData class that represents the modified resource allocation or resource allocation request.

- If the method was executed synchronously and failed completely, the value shall be set to NULL.

6.3.2.4.3 Return codes

An implementation shall indicate the result of the method execution by using the return code values specified in Table 51.

Table 51 – ModifyResourceSettings( ) Method: Return code values

Value	Description
0	Method was successfully executed; all modification requests were successfully processed.
1	Method is not supported.
2	Method execution failed, but some modification requests may have been processed.
3	Method execution failed because a timeout condition occurred, but some modification requests may have been processed.
4	Method execution failed because invalid parameters were specified by the client; no modification requests were processed.
5	Method execution failed because the implementation does not support modifications on virtual resource allocations for the present virtual system state of the virtual system scoping virtual resources affected by this resource allocation modification request.
6	Method execution failed because incompatible parameters were specified by the client; no modification requests were processed.
4096	Method execution is performed asynchronously. The specifications given in 6.3.1.3 apply.
NOTE: Even if the return code indicates a failure, some modification requests may have been successfully executed. In this case, the set of successfully modified resources is conveyed through the value of the ResultingResourceSettings parameter.	

6.3.2.5 CIM_VirtualSystemManagementService.ModifySystemSettings( ) method

The implementation of the ModifySystemSettings( ) method is conditional.

Condition: The modification of virtual systems and resources is implemented; see 6.2.4.6.3.

If the ModifySystemSettings( ) method is implemented, the provisions in this subclause apply; in addition behavior applicable to all extrinsic methods is specified in 6.3.1.2.

The execution of the ModifySystemSettings( ) method shall effect the modification of system settings, such that non-key and non-NULL values of the instance of the CIM_VirtualSystemSettingData class that is provided through the SystemSettings parameter override respective values in the instance identified through the value of the InstanceID property.

Table 52 contains requirements for parameters of this method.

Table 52 – ModifySystemSettings( ) Method: Parameters

Qualifiers	Name	Type	Description/Values
IN	SystemSettings	string	See 6.3.2.5.1.
OUT	Job	CIM_ConcreteJob REF	See 6.3.1.3.2.

6.3.2.5.1 SystemSettings parameter

A client shall set the SystemSettings parameter. Any instance of the CIM_VirtualSystemSettingData class that is passed in as a value of the SystemSettings parameter shall have a valid non-NULL value in the InstanceID property that identifies a respective instance of the CIM_VirtualSystemSettingData class existing within the implementation. A client shall obtain such an instance before invoking the ModifySystemSettings( ) method (for example, by using an extrinsic method or intrinsic CIM operation that yields a respective instance as a result). For example, the client may use the intrinsic GetInstance( ) CIM operation. The client shall then modify the instance locally so that it reflects the desired modifications and finally pass it back in as a value of the SystemSettings parameter.

The implementation shall ignore any value of the SystemSettings parameter that does not identify, through the value of the InstanceID key property, an existing instance of the CIM_VirtualSystemSettingData class within the implementation.

6.3.2.5.2 Return codes

An implementation shall indicate the result of the method execution by using the return code values specified in Table 53.

Table 53 – ModifySystemSettings( ) Method: Return code values

Value	Description
0	Method was successfully executed.
1	Method is not supported.
2	Method execution failed.
3	Method execution failed because a timeout condition occurred.
4	Method execution failed because invalid parameters were specified by the client.
5	Method execution failed because the implementation does not support modifications on virtual system settings for the present virtual system state of the virtual system identified by the input system settings.
6	Method execution failed because incompatible parameters were specified by the client.
4096	Method execution is performed asynchronously. The specifications given in 6.3.1.3 apply.

6.3.2.6 CIM_VirtualSystemManagementService.RemoveResourceSettings( ) method

The implementation of the RemoveResourceSettings( ) method is conditional.

Condition: The addition and the removal of virtual resources to virtual systems is implemented; see 6.2.4.6.2.

If the RemoveResourceSettings( ) method is implemented, the provisions in this subclause apply; in addition behavior applicable to all extrinsic methods is specified in 6.3.1.2.

The execution of the RemoveResourceSettings( ) method shall effect the removal of resource allocation requests identified by the value of elements of the ResourceSettings[ ] parameter.

Table 54 contains requirements for parameters of this method.

Table 54 – RemoveResourceSettings( ) Method: Parameters

Qualifiers	Name	Type	Description/Values
IN	ResourceSettings[ ]	CIM_ResourceAllocationSettingData REF	See 6.3.2.6.1.
OUT	Job	CIM_ConcreteJob REF	See 6.3.1.3.2.

6.3.2.6.1 ResourceSettings[ ] array parameter

A client shall set the ResourceSettings[ ] array parameter. The value of any element specified in the ResourceSettings[ ] array parameter shall represent requests for the removal of virtual resource state extensions, of virtual resource definitions, or both in the scope of one virtual system.

6.3.2.6.2 Return codes

An implementation shall indicate the result of the method execution by using the return code values specified in Table 55.

Table 55 – RemoveResourceSettings( ) Method: Return code values

Value	Description
0	Method execution was successful.
1	Method is not supported.
2	Method execution failed.
3	Method execution failed because a timeout condition occurred.
4	Method execution failed because invalid parameters were specified by the client.
4096	Method execution is performed asynchronously. The specifications given in 6.3.1.3 apply.

6.3.3 Methods of the CIM_VirtualSystemSnapshotService class

This subclause models virtual system snapshot management in terms of methods of the CIM_VirtualSystemSnapshotService class.

6.3.3.1 CIM_VirtualSystemSnapshotService.CreateSnapshot( ) method

The implementation of the CreateSnapshot( ) method is conditional.

Condition: The creation, destruction and application of virtual system snapshots is implemented; see 6.2.7.1.2.

If the CreateSnapshot( ) method is implemented, the provisions in this subclause apply; in addition behavior applicable to all extrinsic methods is specified in 6.3.1.2.

The execution of the CreateSnapshot( ) method shall effect the creation of a snapshot of the affected virtual system. The snapshot shall have the type that is designated by the value of the SnapshotType parameter (see 6.3.3.1.3).

A full snapshot shall contain all information required to restore the complete virtual system and its resources to exactly the situation that existed when the snapshot was created. Other types of snapshots may contain less information.

If the virtual system is in the "Active" virtual system state, it may continue to perform tasks but may be temporarily paused as the creation of the snapshot requires the capturing of state information.

Table 56 contains requirements for parameters of this method.

Table 56 – CreateSnapshot( ) method: Parameters

Qualifiers	Name	Type	Description/Values
IN	AffectedSystem	CIM_ComputerSystem REF	See 6.3.3.1.1.
IN	SnapshotSettings	string	See 6.3.3.1.2.
IN	SnapshotType	uint16	See 6.3.3.1.3.
OUT	ResultingSnapshot	CIM_VirtualSystemSettingData REF	See 6.3.3.1.4.
OUT	Job	CIM_ConcreteJob REF	See 6.3.1.3.2.

6.3.3.1.1 AffectedSystem parameter

A client shall set a value of the AffectedSystem parameter to refer to the instance of the CIM_ComputerSystem class that represents the virtual system that is the source for the snapshot.

An implementation shall interpret the value of the AffectedSystem parameter to identify the virtual system that is the source for the snapshot.

6.3.3.1.2 SnapshotSettings parameter

A client may set a value of the SnapshotSettings parameter with an embedded instance of a CIM_SettingData class. It is assumed that an implementation-specific class derived from CIM_SettingData contains additional implementation-specific properties that enable some control over characteristics of the snapshot process.

An implementation shall use the value of the SnapshotSettings parameter to control the characteristics of the snapshot process.

6.3.3.1.3 SnapshotType parameter

A client shall set the value of the SnapshotType parameter to designate the intended type of snapshot. The value shall be one of the values set in the SnapshotTypesSupported[ ] array property in the instance of the CIM_VirtualSystemSnapshotServiceCapabilities class that is related to the snapshot service.

An implementation shall use the value of the SnapshotType parameter to determine the requested type of snapshot. If a value is not specified or is not one of the values set in the SnapshotTypesSupported[ ] array property in the instance of the CIM_VirtualSystemSnapshotServiceCapabilities class that is related to the snapshot service, an implementation shall fail the method execution and set a return code of 6 (Invalid Type).

6.3.3.1.4 ResultingSnapshot parameter

The implementation shall set the value of the ResultingSnapshot parameter as follows:

- If the method execution is performed synchronously and is successful, the value shall be set to reference the instance of the CIM_VirtualSystemSettingData class that represents the newly created virtual system snapshot.
- If the method execution is performed synchronously and fails, or if the method execution is performed asynchronously, the value shall be set to NULL.

- If the method execution is performed asynchronously and is successful, see 6.3.1.2 to locate the instance of the CIM_VirtualSystemSettingData class that represents the newly created virtual system snapshot.

6.3.3.1.5 Return codes

An implementation shall indicate the result of the method execution by using the return code values specified in Table 57.

Table 57 – CreateSnapshot( ) method: Return code values

Value	Description
0	Method execution was successful.
1	Method is not supported.
2	Method execution failed.
3	Method execution failed because a timeout condition occurred.
4	Method execution failed because an invalid parameter was specified.
5	Method execution failed because the affected system is in a state in which the implementation rejects capturing a snapshot.
6	Method execution failed because no snapshot or an unsupported type of snapshot was requested.
4096	Method execution is performed asynchronously. The specifications given in 6.3.1.3 apply.

6.3.3.2 VirtualSystemSnapshotService.DestroySnapshot( ) method

The implementation of the DestroySnapshot( ) method is conditional.

Condition: Virtual system snapshots are implemented; see 6.2.7.1.2.

If the DestroySnapshot( ) method is implemented, the provisions in this subclause apply; in addition behavior applicable to all extrinsic methods is specified in 6.3.1.2.

The execution of the DestroySnapshot( ) method shall effect the destruction of the affected virtual system snapshot. Dependency relationships from other snapshots to the affected snapshot shall be updated so that the affected snapshot is no longer referenced. If the snapshot was persistently established to be used during virtual system activation, the implementation may assign a different snapshot to be used for subsequent virtual system activations, or may fall back to the "Default" virtual system configuration to be used for future activations. If a virtual system was activated using the snapshot and is still in a state other than the "Defined" virtual system state, the active virtual system shall not be affected by the execution of the DestroySnapshot( ) method.

Table 58 contains requirements for parameters of this method.

Table 58 – DestroySnapshot( ) method: Parameters

Qualifiers	Name	Type	Description/Values
IN	AffectedSnapshot	CIM_VirtualSystemSettingData REF	See 6.3.3.2.1.
OUT	Job	CIM_ConcreteJob REF	See 6.3.1.3.2.

6.3.3.2.1 AffectedSnapshot parameter

A client shall set a value of the AffectedSnapshot parameter to refer to the instance of the CIM_VirtualSystemSettingData class that represents a snapshot.

An implementation shall interpret the value of the AffectedSnapshot parameter to identify the snapshot that is to be destroyed.

6.3.3.2.2 Return codes

An implementation shall indicate the result of the method execution using the return code values specified by Table 59.

Table 59 – DestroySnapshot( ) method: Return code values

Value	Description
0	Method execution was successful.
1	Method is not supported.
2	Method execution failed.
3	Method execution failed because a timeout condition occurred.
4	Method execution failed because an invalid parameter was specified.
5	Method execution failed because the affected snapshot is in a state in which the implementation rejects destroying a snapshot.
6	Method execution failed because the affected snapshot is of a type that is not destroyable.
4096	Method execution is performed asynchronously. The specifications given in 6.3.1.3 apply.

6.3.3.3 VirtualSystemSnapshotService.ApplySnapshot( ) method

The implementation of the ApplySnapshot( ) method is conditional.

Condition: Virtual system snapshots are implemented; see 6.2.7.1.2.

If the ApplySnapshot( ) method is implemented, the provisions in this subclause apply; in addition behavior applicable to all extrinsic methods is specified in 6.3.1.2.

The execution of the ApplySnapshot( ) method shall indicate that the snapshot is used for the next activation of the associated virtual system (the virtual system that was the source for the snapshot). The method execution shall have one or both of the following effects:

- The snapshot is persistently established to be used for subsequent activations.
- The virtual system is immediately activated or recycled, using the snapshot.

Table 60 contains requirements for parameters of this method.

Table 60 – ApplySnapshot( ) method: Parameters

Qualifiers	Name	Type	Description/Values
IN	Snapshot	CIM_VirtualSystemSettingData REF	See 6.3.3.3.1.
OUT	Job	CIM_ConcreteJob REF	See 6.3.1.3.2.

6.3.3.3.1 Snapshot parameter

A client shall set a value of the Snapshot parameter to refer to the instance of the CIM_VirtualSystemSettingData class that represents a snapshot.

An implementation shall interpret the value of the Snapshot parameter to identify the snapshot that is to be applied.

6.3.3.3.2 Return codes

An implementation shall indicate the result of the method execution by using the return code values specified in Table 61.

Table 61 – ApplySnapshot( ) method: Return code values

Value	Description
0	Method execution was successful.
1	Method is not supported.
2	Method execution failed.
3	Method execution failed because a timeout condition occurred.
4	Method execution failed because an invalid parameter was specified.
5	Method execution failed because the affected system is in a state where snapshots cannot be applied.
6	Method execution failed because the type of the affected system does not support the application of a snapshot.
4096	Method execution is performed asynchronously. The specifications given in 6.3.1.3 apply.

6.3.4 Profile conventions for operations

The default list of operations for all classes is:

- GetInstance( )
- EnumerateInstances( )
- EnumerateInstanceNames( )

For classes that are referenced by an association, the default list also includes

- Associators( )
- AssociatorNames( )
- References( )
- ReferenceNames( )

6.3.4.1 CIM_AffectedJobElement

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.2 CIM_ComputerSystem

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.3 CIM_ConcreteJob

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.4 CIM_Dependency

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.5 CIM_ElementCapabilities

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.6 CIM_ElementConformsToProfile

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.7 CIM_HostedDependency

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.8 CIM_HostedService

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.9 CIM_LastAppliedSnapshot

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.10 CIM_MostCurrentSnapshotInBranch

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.11 CIM_ReferencedProfile

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.12 CIM_RegisteredProfile

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.13 CIM_ServiceAffectsElement

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.14 CIM_SnapshotOfVirtualSystem

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.15 CIM_System

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.16 CIM_VirtualSystemManagementCapabilities

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.17 CIM_VirtualSystemManagementService

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.18 CIM_VirtualSystemSnapshotService

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.19 CIM_VirtualSystemSnapshotCapabilities

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.3.4.20 CIM_VirtualSystemSnapshotServiceCapabilities

All operations in the default list in 6.3.4 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

6.4 Use cases

This subclause contains informative text only.

The following use cases and object diagrams illustrate use of the profile described in this clause. They are for informational purposes only and do not introduce behavioral requirements for implementations of the profile.

6.4.1 General assumptions

For all use cases, it is assumed that a client performs intrinsic CIM operations, extrinsic CIM operations, or both.

For all use cases except the use case described in 6.4.2.1, the following conditions are implicitly assumed:

- The client knows the URL of a WBEM service that exposes an implementation of the profile described in this clause.
- The client is able to communicate with the WBEM service through a specified CIM protocol. An example is the use of the http protocol as described in DSP0200. The client may use a facility like a CIM client API to perform the encoding and decoding of CIM messages.

6.4.2 Discovery, localization, and inspection

This set of use cases describes how a client obtains access to an implementation, detects the central and scoped instances, and analyzes information available through these instances. Figure 9 outlines a sample situation that is referenced by some of the use-case descriptions in subsequent subclauses.

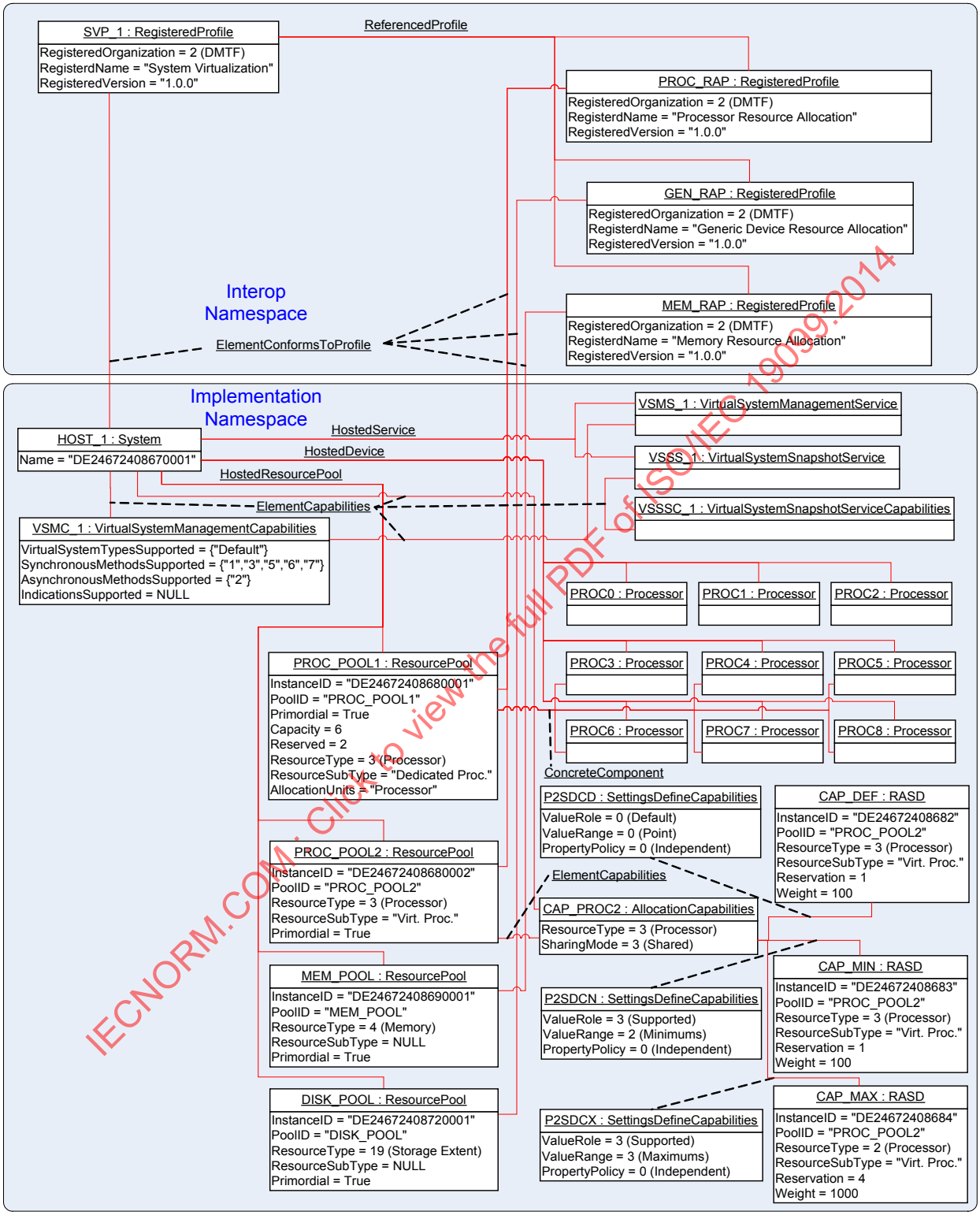

Figure 9 – System Virtualization Profile instance diagram: Discovery, localization, and inspection

6.4.2.1 SLP-Based discovery of CIM object managers hosting implementations of the System Virtualization Profile

The service location protocol (SLP) is used to locate WBEM services. A WBEM service that implements SLP as a discovery mechanism is required to register with SLP all instances of the CIM_RegisteredProfile class that reside in the Interop namespace. An SLP service type is used to identify entities that are registered with SLP. An SLP service type is a structured string variable.

Assumption: The profile described in this clause is registered by at least one WBEM service that maintains a registration with an SLP Directory Agent. The registration includes information about registered DMTF management profiles. The client is able to make SLP calls.

- The client invokes the SLPFindSrvs( ) SLP function as follows:
  - The value of the srvtype parameter is set to "service:wbem".
  - The value of the scopelist parameter is set to "default".
  - The value of the filter parameter is set to "(RegisteredProfilesSupported=DMTF:System Virtualization)".

Result: Each URL in a list of URLs identifies a WBEM service where the profile described in this clause is implemented.

6.4.2.2 Locate conformant implementations using the EnumerateInstances( ) operation

Assumption: The client knows the URL of a WBEM service hosting implementations of the profile described in this clause (see 6.4.2.1).

1) Using the URL, the client invokes the intrinsic EnumerateInstances( ) CIM operation with the value of the ClassName input parameter set to "CIM_RegisteredProfile".

   The result is a list of instances of the CIM_RegisteredProfile class.

2) The client iterates over the list of instances of the CIM_RegisteredProfile class and selects instances where
   - the RegisteredOrganization property has a value of 2 (DMTF)
   - the RegisteredName property has a value of "System Virtualization"
   - the RegisteredVersion property has a value equal to or greater than "1.0.0"

Result: The client knows a set of instances of the CIM_RegisteredProfile class, each representing an implementation of the profile described in this clause.

In the example shown in Figure 9, one instance of the CIM_RegisteredProfile class represents an implementation of the profile described in this clause; it is tagged SVP_1.

6.4.2.3 Locate conformant implementations using the ExecuteQuery( ) operation

Assumption: The client knows the URL of a WBEM service hosting implementations of the profile described in this clause (see 6.4.2.1).

- Using the URL, the client invokes the intrinsic ExecuteQuery( ) CIM operation as follows:
  - The value of the QueryLanguage input parameter is set to "CIM:CQL".
  - The value of the Query input parameter is set to "SELECT * FROM CIM_RegisteredProfile WHERE RegisteredName = ‘System Virtualization’ AND RegisteredVersion >= ‘1.0.0’".

Result: The client knows a set of instances of the CIM_RegisteredProfile class, each representing an implementation of the profile described in this clause.

In the example shown in Figure 9, one instance of the CIM_RegisteredProfile class represents an implementation of the profile described in this clause; it is tagged SVP_1.

6.4.2.4 Locate host systems represented by central instances of the profile described in this clause

Assumption: The client knows a reference to an instance of the CIM_RegisteredProfile class that represents an implementation of the profile described in this clause (see 6.4.2.2 or 6.4.2.3).

- The client invokes the intrinsic AssociatorNames( ) CIM operation as follows:
  - The value of the ObjectName parameter is set to refer to the instance of the CIM_RegisteredProfile class.
  - The value of the AssocClass parameter is set to "CIM_ElementConformsToProfile".
  - The value of the ResultClass parameter is set to "CIM_System".

Result: The client knows a set of references to instances of the CIM_System class that represent host systems that are central and scoping instances of the profile described in this clause.

In the example shown in Figure 9, one instance of the CIM_RegisteredProfile class represents a host system that is a central and scoping instance of the profile described in this clause; it is tagged HOST_1.

6.4.2.5 Locate implementations of scoped resource allocation profiles

Assumption: The client knows a reference to an instance of the CIM_RegisteredProfile class that represents an implementation of the profile described in this clause (see 6.4.2.2 or 6.4.2.3).

1) The client invokes the intrinsic Associators( ) CIM operation to obtain a the list of scoped DMTF management profiles, as follows:
   - The value of the ObjectName parameter is set to refer to the instance of the CIM_RegisteredProfile class.
   - The value of the AssocClass parameter is set to "CIM_ReferencedProfile".
   - The value of the ResultClass parameter is set to "CIM_RegisteredProfile".

   The result is a set of instances of the CIM_RegisteredProfile class that each represent an implementation of a DMTF management profile that is scoped by the profile described in this clause.
2) For each instance of the CIM_RegisteredProfile class, the client determines whether the value of the RegisteredName property matches the registered name of one of the scoped resource allocation DMTF management profiles as specified by Table 43.

   If the value does not match any name of a resource allocation DMTF management profile scoped by the profile described in this clause, the client ignores that instance of the CIM_RegisteredProfile class.

Result: The client knows a set of instances of the CIM_RegisteredProfile class that each represent an implementation of a resource allocation DMTF management profile that is scoped by the profile described in this clause.

In the example shown in Figure 9, three instances of the CIM_RegisteredProfile class are associated with the instance of the CIM_RegisteredProfile class that is tagged SVP_1 and represents a central instance of the profile described in this clause. These instances represent implementations of scoped resource allocation DMTF management profiles:

- The instance tagged PROC_RAP represents an implementation of the Processor Resource Virtualization Profile described in clause 8.

- The instance tagged GEN_RAP represents an implementation of the Generic Device Resource Virtualization Profile described in clause 13.
- The instance tagged MEM_RAP represents an implementation of the Memory Resource Virtualization Profile described in clause 9.

6.4.2.6 Locate virtual system management service

Assumption: The client knows a reference to an instance of the CIM_System class that represents a host system that is a central instance of the profile described in this clause (see 6.4.2.4).

- The client invokes the intrinsic AssociatorNames( ) CIM operation as follows:
  - The value of the ObjectName parameter is set to refer to the instance of the CIM_System class.
  - The value of the AssocClass parameter is set to "CIM_HostedService".
  - The value of the ResultClass parameter is set to "CIM_VirtualSystemManagementService".

Result: The client knows a reference to the instance of the CIM_VirtualSystemManagementService class that represents the virtual system management service that serves the host system. If the operation is successful, the size of the result set is 1.

In the example shown in Figure 9, one instance of the CIM_VirtualSystemManagementService class serves the host system; it is tagged VSMS_1.

6.4.2.7 Determine the capabilities of an implementation

Assumption: The client knows a reference to an instance of the CIM_System class that represents a host system that is a central instance of the profile described in this clause (see 6.4.2.4).

1) The client invokes the intrinsic Associators( ) CIM operation as follows:
   - The value of the ObjectName parameter is set to refer to the instance of the CIM_System class.
   - The value of the AssocClass parameter is set to "CIM_ElementCapabilities".
   - The value of the ResultClass parameter is set to "CIM_VirtualSystemManagementCapabilities".

   The result is a list of instances of the CIM_VirtualSystemManagementCapabilities class. If the operation is successful, the size of the result set is 1.
2) The client analyzes the instance of the CIM_VirtualSystemManagementCapabilities class.
   - The VirtualSystemTypesSupported[ ] array property lists identifiers of virtual system types that the implementation supports.
   - The SynchronousMethodsSupported[ ] array property lists identifiers of methods of the CIM_VirtualSystemManagementService class that are implemented with synchronous method execution only.
   - The AsynchronousMethodsSupported[ ] array property lists identifiers of methods of the CIM_VirtualSystemManagementService class that are implemented with synchronous and asynchronous method execution.
   - The IndicationsSupported[ ] array property lists identifiers of types of indications that the implementation supports.

Result: The client knows the capabilities of the host system in terms of properties of the CIM_VirtualSystemManagementCapabilities class.

In the example shown in Figure 9, one instance of the CIM_VirtualSystemManagementCapabilities class is associated with the host system; it is tagged VSMC_1.

- The VirtualSystemTypesSupported[ ] array property lists one element with the value "Default", which indicates that the implementation supports one virtual system type named "Default". The semantics are implementation specific.
- The SynchronousMethodsSupported[ ] array property lists enumerated values: { 1 (AddResourceSettingsSupported), 3 (DestroySystemSupported), 5 (ModifyResourceSettingsSupported), 6 (ModifySystemSettingsSupported), and 7 (RemoveResourcesSupported) }, which indicates that the AddResources( ) method, the DestroySystem( ) method, the ModifyResourceSettings( ) method, and the RemoveResourceSettings( ) method are implemented by the implementation with synchronous execution.
- The AsynchronousMethodsSupported[ ] array property lists the enumerated value { 2 (DefineSystemSupported) }, which indicates that the DefineSystem( ) method is implemented by the implementation with synchronous or asynchronous execution.
- The value of the IndicationsSupported[ ] array property is NULL, which indicates that indications are not implemented by the implementation.

6.4.2.8 Locate hosted resource pools of a particular resource type

Assumption: The client knows a reference to an instance of the CIM_System class that represents a host system that is a central instance of the profile described in this clause (see 6.4.2.4).

1) The client invokes the intrinsic Associators( ) CIM operation as follows:
   - The value of the ObjectName parameter is set to refer to the instance of the CIM_System class.
   - The value of the AssocClass parameter is set to "CIM_HostedResourcePool".
   - The value of the ResultClass parameter is set to "CIM_ResourcePool".

   The result is a list of instances of the CIM_ResourcePool class.
2) For each instance of CIM_ResourcePool, the client determines whether the value of the ResourceType property matches the requested resource type.

   If the value does not match the requested resource type, the client drops that instance of the CIM_ResourcePool class from the list.

Result: The client knows a set of instances of the CIM_ResourcePool class, each representing a hosted resource pool of the requested resource type.

6.4.2.9 Obtain a set of central instances of scoped resource allocation profiles

Resource allocation DMTF management profiles are based on the Resource Allocation Profile (see clause 5), which defines the CIM_ResourcePool class as the central class. The procedure for the determination of central instances of scoped DMTF management profiles depends on the profile advertisement methodology applied by the respective implementations.

Assumption: The client knows a reference to an instance of the CIM_RegisteredProfile class that represents an implementation of a scoped DMTF management profile (see 6.4.2.5).

- The client invokes the intrinsic Associators( ) CIM operation to obtain the list of instances of the CIM_ResourcePool class that are central instances of the scoped DMTF management profiles, as follows:

- The value of the ObjectName parameter is set to refer to the instance of the CIM_RegisteredProfile class
- The value of the AssocClass parameter is set to "CIM_ElementConformsToProfile".
- The value of the ResultClass parameter is set to "CIM_ResourcePool".

The result is a list of instances of the CIM_ResourcePool class; the list may be empty.

- If the list is not empty, the *central class* profile implementation advertisement methodology is applied by the implementation for the scoped resource allocation DMTF management profile. In this case, the list is the result for this use case.
- If the list is empty, the *scoping class* profile implementation advertisement methodology is applied by the implementation for the scoped resource allocation DMTF management profile. In this case, the client
  - needs to know the resource type associated with the scoped resource allocation DMTF management profile
  - applies use case 6.4.2.8 to obtain a list of instances of the CIM_ResourcePool class that each represent a resource pool of that particular resource type.

  The resulting list is the result for this use case.

Result: The client knows a list of instances of the CIM_ResourcePool class, each representing a central instance of a scoped resource allocation DMTF management profile.

6.4.2.10 Determine implemented resource types

Assumption: The client knows a reference to an instance of the CIM_RegisteredProfile class that represents an implementation of the profile described in this clause (see 6.4.2.2 or 6.4.2.3).

1) The client locates implementations of DMTF management profiles that are scoped by the profile described in this clause (see 6.4.2.5).

   The result is a list of references to instances of the CIM_RegisteredProfile class that represent implementations of DMTF management profiles that are scoped by the profile described in this clause.

2) For each instance of CIM_RegisteredProfile, the client obtains the set of instances of the CIM_ResourcePool class that are central instances of the respective scoped resource allocation DMTF management profiles and represent a conformant resource pool (see 6.4.2.9).

   The result is a list of instances of the CIM_ResourcePool class that are central instances of scoped resource allocation DMTF management profiles.

3) The client creates an initially empty list of integer values. For each instance that is a result from step 2), the client determines whether the value of property ResourceType is already represented in the list:
   - If that value is already contained in the list, the client ignores the element.
   - If that value is not yet contained in the list, the client adds a new element to the list with that value.

Result: The client knows a list of integer values, each designating a resource type that is supported by the implementation.

In the example shown in Figure 9, three instances of the CIM_RegisteredProfile class are associated with the instance of the CIM_RegisteredProfile class that represents the implementation of the profile described in this clause. These instances are central instances of scoped resource allocation DMTF management profiles:

- The instance tagged PROC_RAP represents an implementation of the Processor Resource Virtualization Profile described in clause 8.
- The instance tagged GEN_RAP represents an implementation of the Generic Device Resource Virtualization Profile described in clause 13.
- The instance tagged MEM_RAP represents an implementation of the Memory Resource Virtualization Profile described in clause 9.

These instances are all associated with respective instances of the CIM_ResourcePool class, indicating that in this example in all cases the central class profile advertisement methodology is in use:

- The instance tagged PROC_RAP is associated with two instances that represent resource pools for the allocation of processors. They show a value of 3 (Processor) for the ResourceType property and are tagged PROC_POOL1 and PROC_POOL2.
- The instance tagged GEN_RAP is associated with one instance that represents a resource pool for the allocation of virtual disks. It shows a value of 19 (Storage Extent) for the ResourceType property and is tagged DISK_POOL.
- The instance tagged MEM_RAP is associated with one instance that represents a resource pool for the allocation of memory. It shows a value of 4 (Memory) for the ResourceType property and is tagged MEM_POOL.

The resulting list of integer values is {"3","4","19"} and designates the implemented resource types 3 (Processor), 4 (Memory), and 19 (Storage Extent).

6.4.2.11 Determine the default resource pool for a resource type

Assumption: The client knows a reference to an instance of the CIM_System class that represents a host system that is a central instance of the profile described in this clause (see 6.4.2.4).

1) The client invokes the intrinsic Associators( ) CIM operation for a list of allocation capabilities associated with resource pools hosted by the host system, as follows:
   - The value of the ObjectName parameter is set to refer to the instance of the CIM_System class.
   - The value of the AssocClass parameter is set to "CIM_ElementCapabilities".
   - The value of the ResultClass parameter is set to "CIM_AllocationCapabilities".

   The result is a list of instances of the CIM_AllocationCapabilities class.
2) The client drops instances from the result list of step 1) that have a value for the ResourceType property that does not match the requested resource type.

   The purpose of the following two steps is to further limit the result set from step 2) to those instances of the CIM_AllocationCapabilities class that describe default settings. Default settings are flagged in the connecting instance of the CIM_ElementCapabilities association that has a value of 2 (Default) for the Characteristics property.
3) For each instance of the list resulting from step 2), the client invokes the intrinsic References( ) CIM operation for a list of association instances that refer to the resource pool:
   - The value of the ObjectName parameter refers the instance of the CIM_ResourcePool class.
   - The value of the ResultClass parameter is set to "CIM_AllocationCapabilities".

   The result is a list of instances of the CIM_ElementCapabilities association that associate an instance of the CIM_ResourcePool class that is taken from the result of step 2).

4) From the list obtained in step 3), the client drops all elements that meet either of the following conditions:

   – have a value other than 2 (Default) for the Characteristics property

   – do not refer to the instance of the CIM_System class that represents the host system through the ManagedElement property

   The list should now contain one instance of the CIM_AllocationCapabilities class that represents default allocation capabilities for the resource type in question.

5) The client invokes the intrinsic Associators( ) CIM operation to resolve association for the resource pool, as follows:

   – The value of the ObjectName parameter refers to the instance of the CIM_AllocationCapabilities class selected in step 4).

   – The value of the AssocClass parameter is set to "CIM_ElementCapabilities".

   – The value of the ResultClass parameter is set to "CIM_ResourcePool".

   The result is a list of instances of the CIM_ResourcePool class. The size of the list is 1.

Result: The client knows the instance of the CIM_ResourcePool class that represents the default resource pool for the requested resource type.

In the example shown in Figure 9, allocation capabilities are depicted only for the virtual processor pool. In the subsequent description, it is assumed that the client looks for the default resource pool for processors:

- With step 1) of this use case, the client resolves the CIM_ElementCapabilities association from the instance of the CIM_System class that represents the host system (tagged HOST_1) to instances of the CIM_AllocationCapabilities class. A conformant implementation of the Allocation Capabilities Profile (described in clause 7) shows only one associated element for each resource type.
- With step 2), the client reduces the result set to the one element that describes allocation capabilities processors. This instance is tagged CAP_PROC1.
- With steps 3) and 4), the client further reduces the result set to the one instance of the CIM_AllocationCapabilities class that represents the system's default capabilities for resource type 3 (Processor).
- With step 5), the client resolves the CIM_ElementCapabilities association in order to obtain the instance of the CIM_ResourcePool class that represents the default resource pool for processors. This instance is tagged PROC_POOL2.

6.4.2.12 Determine the resource pool for a resource allocation request or an allocated resource

Assumption: The client knows a reference to an instance of the CIM_ResourceAllocationSettingData class that represents a resource allocation request or allocated resource.

- The client invokes the intrinsic Associators( ) CIM operation for a list of allocation capabilities associated with resource pools hosted by the host system, as follows:

  – The value of the ObjectName parameter is set to refer to the instance of the CIM_ResourceAllocationSettingData class.

  – The value of the AssocClass parameter is set to "CIM_ResourceAllocationFromPool".

  – The value of the ResultClass parameter is set to "CIM_ResourcePool".

  The result is a list of instances of the CIM_ResourcePool class containing one element.

Result: The client knows the instance of the CIM_ResourcePool class that represents the resource pool for the resource allocation request or allocated resource.

6.4.2.13 Determine valid settings for a resource type

This use case describes the determination of valid settings for a resource type in the context of either the system as a whole or one resource pool.

Assumption: The client knows a reference to either of the following instances:

- an instance of the CIM_ResourcePool class that represents a resource pool that is a central instance of a resource allocation DMTF management profile
- an instance of the CIM_System class that represents a host system

The sequence of activities is as follows:

1) The client invokes the intrinsic Associators( ) CIM operation as follows:
   - The value of the ObjectName parameter is set to refer to the instance of the CIM_ResourcePool class or the CIM_System class.
   - The value of the AssocClass parameter is set to "CIM_ElementCapabilities".
   - The value of the ResultClass parameter is set to "CIM_AllocationCapabilities".

   The result is a list of instances of the CIM_AllocationCapabilities class that describe the capabilities of the input instance.
2) The client drops from the result of step 1) those instances in which the ResourceType property designates a resource type other than the requested resource type. This step is required only if the starting point of the use case was an instance of the CIM_System class.

   At this point the client has a list of instances of the CIM_AllocationCapabilities class that describe allocation capabilities. The value of the SharingMode property allows a distinction between shared and dedicated resources.
3) The client invokes the intrinsic References( ) CIM operation for a set of instances of the CIM_SettingsDefineCapabilities association that each associate one instance of the CIM_ResourceAllocationSettingData class that describes a limiting aspect (min/max/increment), as follows:
   - The value of the ObjectName parameter is set to refer to the instance of the CIM_AllocationCapabilities class.
   - The value of the ResultClass parameter is set to "CIM_SettingsDefineCapabilities".

   The result is a list of instances of the CIM_SettingsDefineCapabilities association.
4) For each instance that is a result from step 3), the client analyzes the values of the PropertyPolicy property and the ValueRange property. The value of the ValueRole property is irrelevant in this case.

   The property values have the following impact:
   - The value of the PropertyPolicy property is 0 (Independent) for a conformant implementation of the Allocation Capabilities Profile (described in clause 7) in association instances that connect a min/max/increment limiting setting.
   - The value of the ValueRange property allows determining the designation of the associated setting:

- A value of 1 (Minimums) indicates that the referenced instance of the CIM_ResourceAllocationSettingData class represents a lower limit for the allocation of resources of the respective resource type.
- A value of 2 (Maximums) indicates that the referenced instance of the CIM_ResourceAllocationSettingData class represents an upper limit for the allocation of resources of the respective resource type.
- A value of 3 (Increments) indicates that the referenced instance of the CIM_ResourceAllocationSettingData class represents an increment for the allocation of resources of the respective resource type.

5) For each association instance obtained in step 4), the client invokes the intrinsic GetInstance( ) CIM operation for the instance of the CIM_ResourceAllocationSettingData class that describes the respective limitation. The value of InstanceName parameter is set to the value of the PartComponent property in the association instance obtained in step 4).

   In each case, the result is an instance of the CIM_ResourceAllocationSettingData class that represents a limiting setting.

Result: The client knows the valid resource settings for the requested resource type.

6.4.2.14 Determine implementation class specifics

The profile described in this clause specifies the use of classes derived from the CIM_SettingData class, namely the CIM_VirtualSystemSettingData class and the CIM_ResourceAllocationSettingData class. Instances of these classes are used to describe requirements on virtual systems and virtual resources as these are created or modified. An implementation may provide platform-specific implementation classes that extend these classes (or, for the CIM_ResourceAllocationSettingData class, that extend resource-type-specific extensions specified in a resource-type-specific resource allocation DMTF management profile).

A client should be prepared to deal with these extensions. A client should obtain class information for all derived classes it deals with, in particular focusing on all class qualifiers and all property qualifiers, namely

- the Description qualifier that provides a description of the subclass or property
- the DisplayName qualifier that provides a name for each subclass or property that is potentially known to end-users

Assumption: The client knows a reference to an instance of the class for which the client wants to obtain class-specific information.

1) The client extracts the class name from the reference.
2) The client invokes the intrinsic GetClass( ) CIM operation to obtain a formal class description, as follows:
   - The value of the ClassName parameter is set to the name of the class.
   - The value of the LocalOnly parameter is set to "false".
   - The value of the IncludeQualifiers parameter is set to "true".
   - The value of the IncludeClassOrigin parameter is set to "true".

   The result is a description of a CIM class.

Result: The client has a description of the class. The format depends on the CIM client used to issue the request and is based on the XML class data structure that describes a CIM class as defined in DSP0201. The description contains the class's qualifiers, its properties with property qualifiers, and its methods with

method qualifiers. Inspection of the class description enables the client to create local instances of the respective implementation class.

6.4.2.15 Determine the implementation class for a resource type

Assumption: The client knows a list of references to instances of the CIM_ResourcePool class that represent resource pools available at a host system.

1) The client applies use case 6.4.2.13 to obtain a reference to an instance of the CIM_ResourceAllocationSettingData class that is associated with an instance of the CIM_ResourcePool class of the requested type through an instance of the CIM_SettingsDefineCapabilities association with the ValueRole property set to "DEFAULT".
2) The client applies use case 6.4.2.14 to obtain class information about that instance.

Result: The client has an implementation class descriptor, which allows the client to analyze the implementation class for its qualifiers, its properties and their qualifiers, and its methods and their qualifiers. Further, the client can create local instances of the returned class that may be used as input on methods of the CIM_VirtualSystemManagementService class.

6.4.2.16 Locate virtual systems hosted by a host system

Assumption: The client knows a reference to an instance of the CIM_System class that is the central instance of the profile described in this clause and represents a host system (see 6.4.2.4).

- The client invokes the intrinsic AssociatorNames( ) CIM operation for the list of virtual systems, as follows:
  - The value of the ObjectName parameter is set to refer to the instance of the CIM_System class.
  - The value of the AssocClass parameter is set to "CIM_HostedSystem".
  - The value of the ResultClass parameter is set to "CIM_ComputerSystem".

  The result is a list of references to instances of the CIM_ComputerSystem class.

Result: The client knows a set of references to instances of the CIM_ComputerSystem class that represent virtual systems that are hosted by the host system.

6.4.3 Virtual system definition, modification, and destruction

General assumption: The client knows a reference to an instance of the CIM_VirtualSystemManagementService class that represents the virtual system management services of a host system (see 6.4.2.6).

6.4.3.1 Virtual system definition

Virtual system definition is performed using a client-provided configuration, a configuration of an existing virtual system, a configuration that is stored within the implementation, or combinations of these.

6.4.3.1.1 Define virtual system based on input and reference virtual system configuration

Assumption: No assumption is made beyond the general assumption specified in 6.4.3.

1) The client invokes the DefineSystem( ) method (see 6.3.2.1) on the virtual system management service, as follows.
   - The value of the SystemSettings parameter is set to an embedded instance of the CIM_VirtualSystemSettingData class.

- The value of the ResourceSettings[ ] array parameter is set to an array of embedded instances of the CIM_ResourceAllocationSettingData class.
- The value of the ReferenceConfiguration parameter is set to refer to a "Reference" virtual system configuration.

2) The implementation executes the DefineSystem( ) method. The configuration of the new virtual system is created according to the client's requirements. The new virtual system is in the "Defined" virtual system state.

   The value returned in the ResultingSystem parameter refers to an instance of the CIM_ComputerSystem class.

Result: The client knows a reference to an instance of the CIM_ComputerSystem class that represents the new virtual system.

Figure 10 shows the representation of a virtual system that was defined using an "Input" virtual system and a "Reference" virtual system configuration.

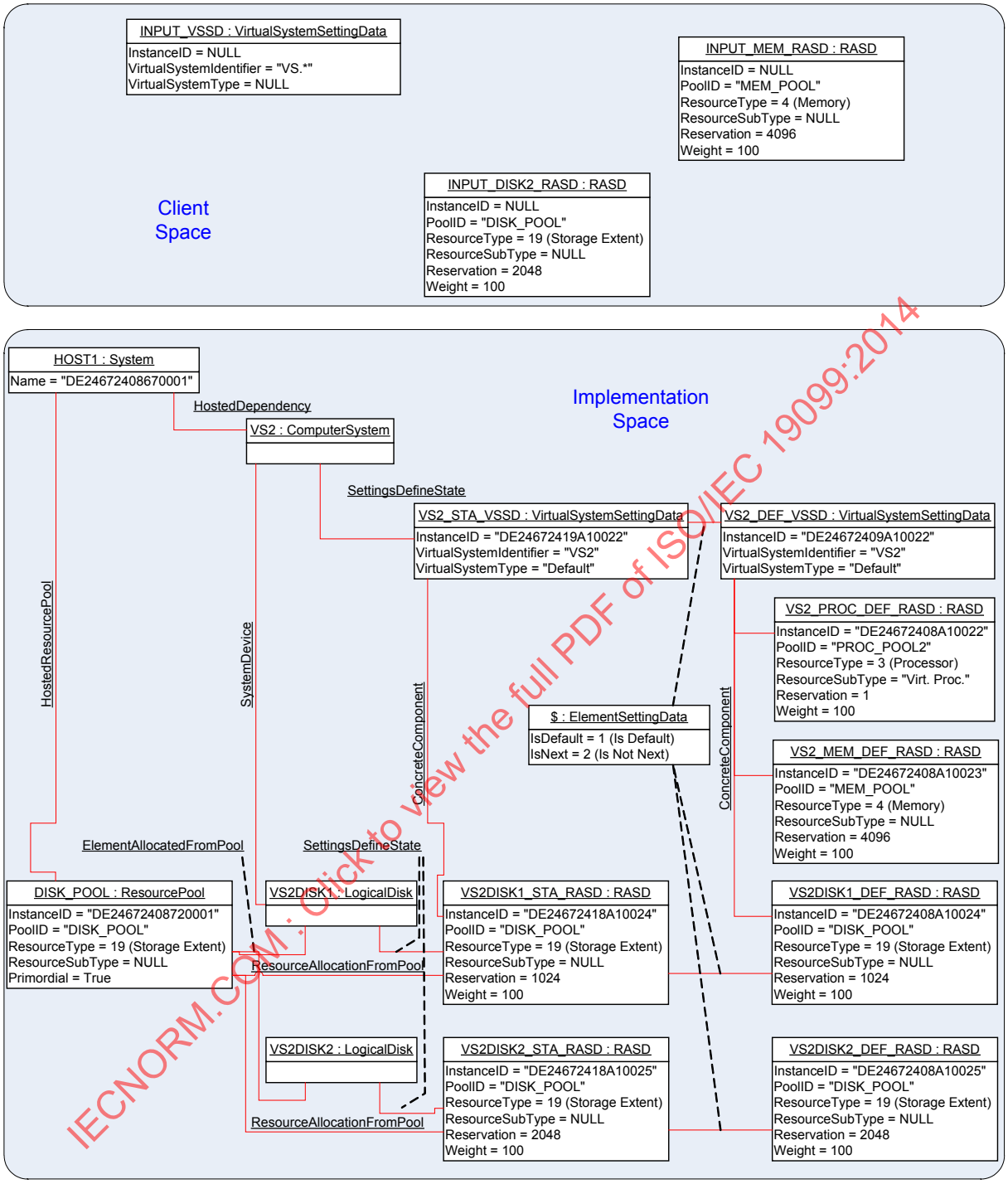

Figure 10 – Virtual system configuration based on input virtual system configurations and implementation defaults

The new virtual system is represented by an instance of the CIM_ComputerSystem class that is tagged VS2. The right side of Figure 10 shows the "Defined" virtual system configuration for the new virtual system. It is based on the "Input" virtual system configuration shown at the top of Figure 10. In this example, it is assumed that the ReferenceConfiguration parameter refers to a virtual system configuration that contains requests for the following resources:

- a virtual processor
- virtual memory of 1024 MB
- a virtual disk of 1024 MB

The "Input" virtual system configuration does not request the allocation of a processor, but because the "Reference" virtual configuration does, the resulting virtual system definition contains a request for a processor as well.

The input virtual system configuration requests 4096 MB of memory. That value is given preference over the value of 1024 that is specified in the "Reference" configuration.

The input virtual system configuration requests a virtual disk in addition to the one requested by the "Reference" configuration, resulting in two virtual disks allocated for the new virtual system.

6.4.3.1.2 Define virtual system with implementation-specific properties

Assumption: No assumption is made beyond the general assumption specified in 6.4.3.

- The client performs use case 6.4.3.1.1 using an input configuration only. While preparing the input virtual system configuration, the client applies use case 6.4.2.14 to determine the implementation class of the CIM_VirtualSystemSettingData class and use case 6.4.2.15 to determine the various implementation classes for the CIM_ResourceAllocationSettingData class for the required resource types.

  The implementation classes may specify additional properties beyond the set that is defined in the respective base classes. The client may use the description information about each of these properties that is obtained with the respective class descriptions to request appropriate values from end users in order to create valid instances of the implementation class (thereby defining implementation-specific resource requirements).

Result: The value of the DefinedSystem output parameter refers to an instance of the CIM_ComputerSystem class that represents the newly created virtual system. The new system is in the "Defined" state.

6.4.3.2 Virtual system modification

This subclause describes a set of usecases that modify virtual systems or virtual system configurations.

6.4.3.2.1 Modify virtual system state or definition

Assumption: The client knows a reference to an instance of the CIM_ComputerSystem class that represents a virtual system.

1) The client obtains the instance of the CIM_VirtualSystemSettingData class that represents the state or definition of virtual aspects of the affected virtual system. (Use cases for the Virtual System Profile are described in 12.4.)
2) The client makes conformant changes to the instance of the CIM_VirtualSystemSettingData class. In particular, the client must not modify key properties.
3) The client invokes the ModifySystemSettings( ) method (see 6.3.2.5) on the virtual system management service. The value of the SystemSettings parameter is the modified instance from step 2).

4) The implementation executes the ModifySystemSettings( ) method, and the configuration of the virtual system is modified according to the clients requirements.

Result: The requested modification is applied to the state or definition of the virtual system.

6.4.3.2.2 Add virtual resources

Assumption: The client knows a reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system configuration.

1) The client locally prepares one or more instances of the CIM_ResourceAllocationSettingData class to represent the resource allocation requests for the new virtual resources.

2) The client invokes the AddResourceSettings( ) method (see 6.3.2.3) on the virtual system management service, as follows:

   – The value of the AffectedConfiguration parameter is set to refer to the instance of the CIM_VirtualSystemSettingData class that represents the virtual system configuration that receives new resources allocations.

   – The value of the ResourceSettings[ ] array parameter is set with each element as one embedded instance of the CIM_ResourceAllocationSettingData class prepared in step 1).

3) The implementation executes the AddResourceSettings( ) method, adding the requested resource allocations and resource allocation requests to the virtual system configuration.

Result: The requested resource allocations or resource allocation requests are configured into the referenced virtual system configuration.

6.4.3.2.3 Modify virtual resource state extension or virtual resource definition

Assumption: The client knows references to one or more instances of the CIM_LogicalDevice class that represent one or more virtual resources.

Alternatively the client knows the reference to an instance of the CIM_ResourceAllocationSettingData class that represents the virtual resource state extensions or virtual resource definitions. In this case, the client would obtain the referenced instance by using the intrinsic GetInstance( ) CIM operation and proceed with step 4).

1) The client invokes the intrinsic Associators( ) CIM operation for the virtual resource state extension as follows:

   – The value of the ObjectName parameter is set to refer to the instance of the CIM_LogicalDevice class.

   – The value of the AssocClass parameter is set to "CIM_SettingsDefineState".

   – The value of the ResultClass parameter is set to "CIM_ResourceAllocationSettingData".

   The result is a list of instances of the CIM_ResourceAllocationSettingData class. The size of the list is expected to be 1, and that element represents the virtual resource state extension. If the client intends to modify the virtual resource state extension, the client skips steps 2) and 3), and proceeds with step 4). If the client intends to modify the virtual resource definition, the client continues with step 2).

2) The client invokes the intrinsic References( ) CIM operation for the association instances that connect the virtual resource definition, as follows:

   – The value of the ObjectName parameter is set to refer to the instance of the CIM_ResourceAllocationSettingData class that was obtained in step 1).

   – The value of the ResultClass parameter is set to "CIM_ElementSettingData".

The result is a list of instances of the CIM_ElementSettingData association that connect various settings to the virtual resource state extension.

3) The client selects from the result set of step 2) the instance in which the IsDefault property has a value of 1 (Is Default). In that instance, the value of the SettingData property refers to the instance of the CIM_ResourceAllocationSettingData class that represents the virtual resource definition.

4) The client invokes the intrinsic GetInstance( ) CIM operation for the setting that represents the resource allocation definition. The value of the InstanceName parameter is set to the value of the SettingData property from the instance of the CIM_ElementSettingData association selected in step 3).

   The result is the instance of the CIM_ResourceAllocationSettingData class that represents the virtual resource definition.

5) The client makes conformant changes to the instance of the CIM_ResourceAllocationSettingData class. In particular, the client must not modify key properties.

   Eventually the client executes steps 1) to 5) repetitively, preparing a set of resource allocation change requests that subsequently are applied as one atomic operation.

6) The client invokes the ModifyResourceSettings( ) method (see 6.3.2.4) on the virtual system management service. The values of elements of the ResourceSettings parameter are the modified instances of the CIM_ResourceAllocationSettingData class that were prepared through repetitive execution of steps in steps 1) to 5).

7) The implementation executes the ModifyResourceSettings( ) method, causing the requested resource allocation changes being applied to resource allocation state extensions or resource allocation definitions.

Result: The requested resource modifications are applied to virtual resource state extensions or virtual resource definitions.

Figure 11 shows the representation of a virtual system. Initially the virtual system was instantiated according to the "Defined" virtual system configuration that is show on the right side. During the activation of the virtual system, required resources were allocated. Virtual resources are represented by instances of subclasses of the CIM_LogicalDevice class (CIM_Processor, CIM_Memory, or CIM_LogicalDisk in this case), with their "State" extensions in the "State" virtual system configuration. Related elements in the virtual system representation and the "State" virtual system configuration are associated through instances of the CIM_SettingsDefineState association.

Entities that are shown in blue color in Figure 11 are involved in the example of a processor resource modification that is described following the figure.

Next, the client applied a resource modification on the allocated processor resource within the virtual system's "State" configuration. The "State" configuration is shown to the left of the "Defined" virtual system configuration. The client obtained a local copy of the instance of the CIM_ResourceAllocationSettingData class that is tagged VS2_PROC_STA_RASD. In that local copy, the client modified the value of the Reservation property to 2 and the value of the Weight property to 200. Then the client called the ModifyResourceSettings( ) method with the modified instance as the only element value for the ResourceSettings[ ] array parameter. The execution of that method resulted in another virtual processor being allocated to the virtual system.

NOTE: Because a change applied to the "State" virtual system configuration is temporary in nature, a recycling of the virtual system will nullify the change and result in a new "State" virtual system configuration based on the "Defined" virtual system configuration.

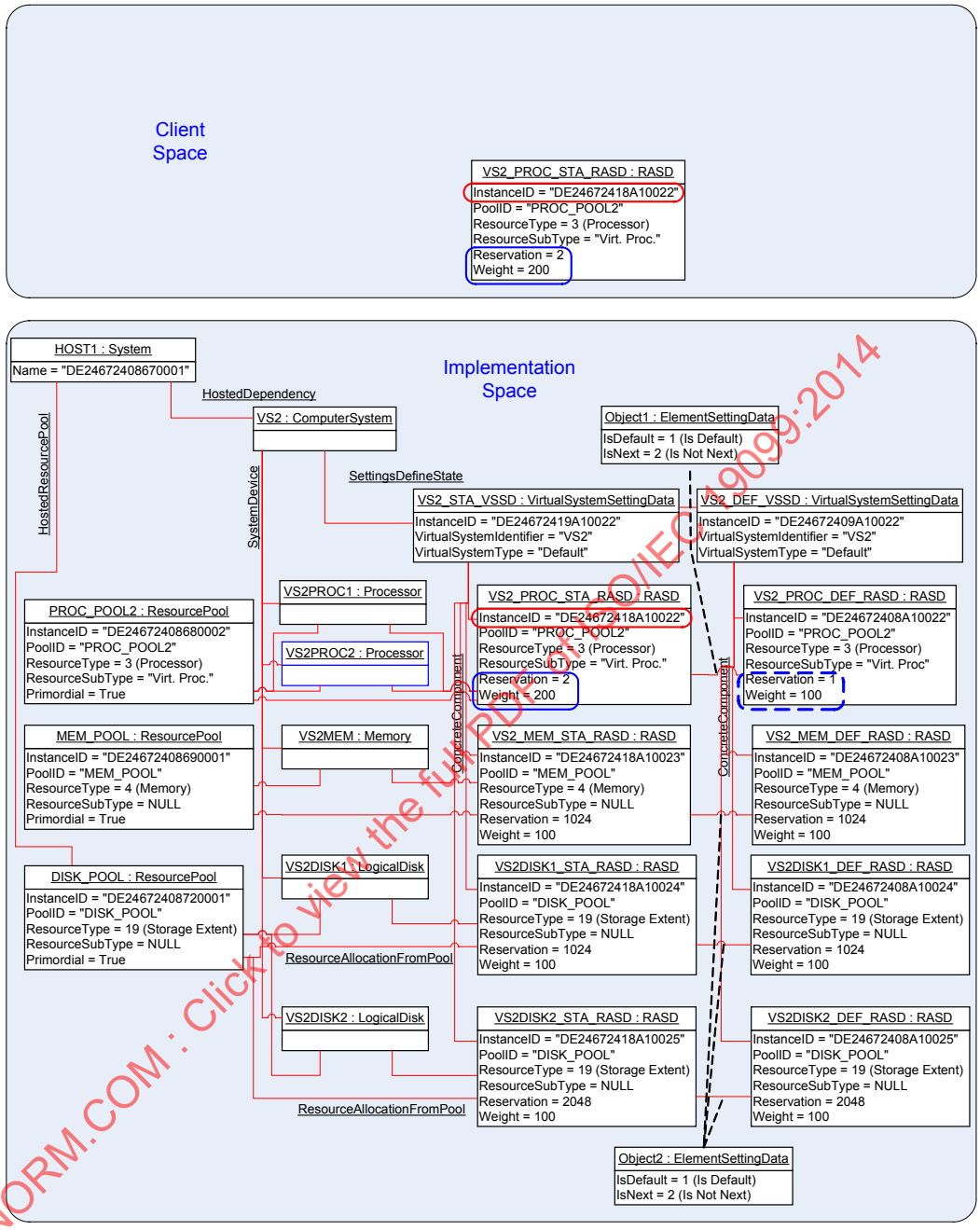

Figure 11 – Virtual system resource modification

6.4.3.2.4 Delete virtual resources or virtual resource definitions

Assumption: The client has references to one or more instances of the CIM_ResourceAllocationSettingData class that refer to elements of the "State" or "Defined" virtual system configuration of one virtual system. Respective use cases for the Virtual System Profile are described in 12.4.

1) The client invokes the RemoveResourceSettings( ) method (see 6.3.2.6) on the virtual system management service. The value of the ResourceSettings[ ] array parameter is set with each element referring to one instance of the CIM_ResourceAllocationSettingData class.

2) The implementation executes the RemoveResourceSettings( ) method. Either all requested resource allocations or resource allocation requests are removed, or none at all.

Result: The referenced virtual resources are removed from their respective virtual system configurations.

6.4.3.3 Destroy virtual system

Assumption: The client knows a reference to an instance of the CIM_ComputerSystem class that represents a virtual system (see 6.4.2.16).

1) The client invokes the DestroySystem( ) method on the virtual system management service. The value of the AffectedSystem parameter is set to refer to the instance of the CIM_ComputerSystem class that represents the virtual system.

2) The implementation executes the DestroySystem( ) method.

Result: The affected virtual system and its virtual resources (together with their definition) are removed from the implementation. If the virtual system was in the "Active" state, the "Paused" state, or in the "Suspended" state, the running instance of the virtual system and its virtual resources are removed before the definition of the virtual system is removed.

NOTE Dependencies may exist that may prevent the destruction of a virtual system. For example, if definitions or instances of other virtual systems refer to elements of the virtual system to be destroyed, the destruction may fail.

6.4.4 Snapshot-related activities

This set of use cases describes activities such as the following:

- discovering a virtual system snapshot service
- inspecting the capabilities of a virtual system snapshot service
- creating a snapshot from a virtual system
- applying a snapshot to a virtual system
- analyzing a virtual snapshot
- analyzing dependencies among snapshots
- locating the most recently captured snapshot
- destroying a snapshot

Figure 12 depicts the CIM representation of a virtual system VS1 and of configurations that are associated with the virtual system at time T3. In the example, it is assumed that the implementation applies the "Single-Configuration Implementation Approach" as described in the Virtual System Profile in clause 12.

The sequence of events that yield the situation shown in Figure 12 is as follows:

1) At time T0, the virtual system VS1 is defined. The initial virtual system definition contains virtual resource allocation requests for one memory extent, one virtual processor, and one virtual disk.

2) At a time after T0 but before T1, the virtual system is activated.

3) At time T1, a full snapshot S1 is captured of the virtual system. Virtual system definition and state are copied into the snapshot. A full snapshot includes the "content" of virtual memory *and* of virtual disks; a disk snapshot would contain the "content" of virtual disks only.

4) The virtual system remains active after the snapshot is captured. The virtual system configuration and the "content" of memory and of virtual disks may change in that interval.

5) At a time after T1 but before T2, snapshot S1 is applied to the virtual system, causing definition and state to be restored to the situation at time T1.

6) Still at a time before T2, a second virtual disk is dynamically added to the virtual system. Because in this example the implementation applies the "Single-Configuration Implementation Approach," this change in effect applies to both virtual system definition and virtual system instance and is visible through the "Single" VS configuration.

7) At time T2, snapshot S2 is captured of the virtual system. Because at time T2 the virtual system snapshot S1 is the last applied snapshot, snapshot S2 depends on snapshot S1.

8) The virtual system remains active after the snapshot is captured. The virtual system configuration and the "content" of memory and of virtual disks may change in that interval.

9) At a time after T2 but before T3, snapshot S2 is applied to the virtual system, causing definition and state to be restored to the situation at time T2, thereby nullifying changes that were applied to the virtual system after T2.

10) At time T3, the situation is as shown in Figure 12.

Current Time: T3 > T2 > T1 > T0

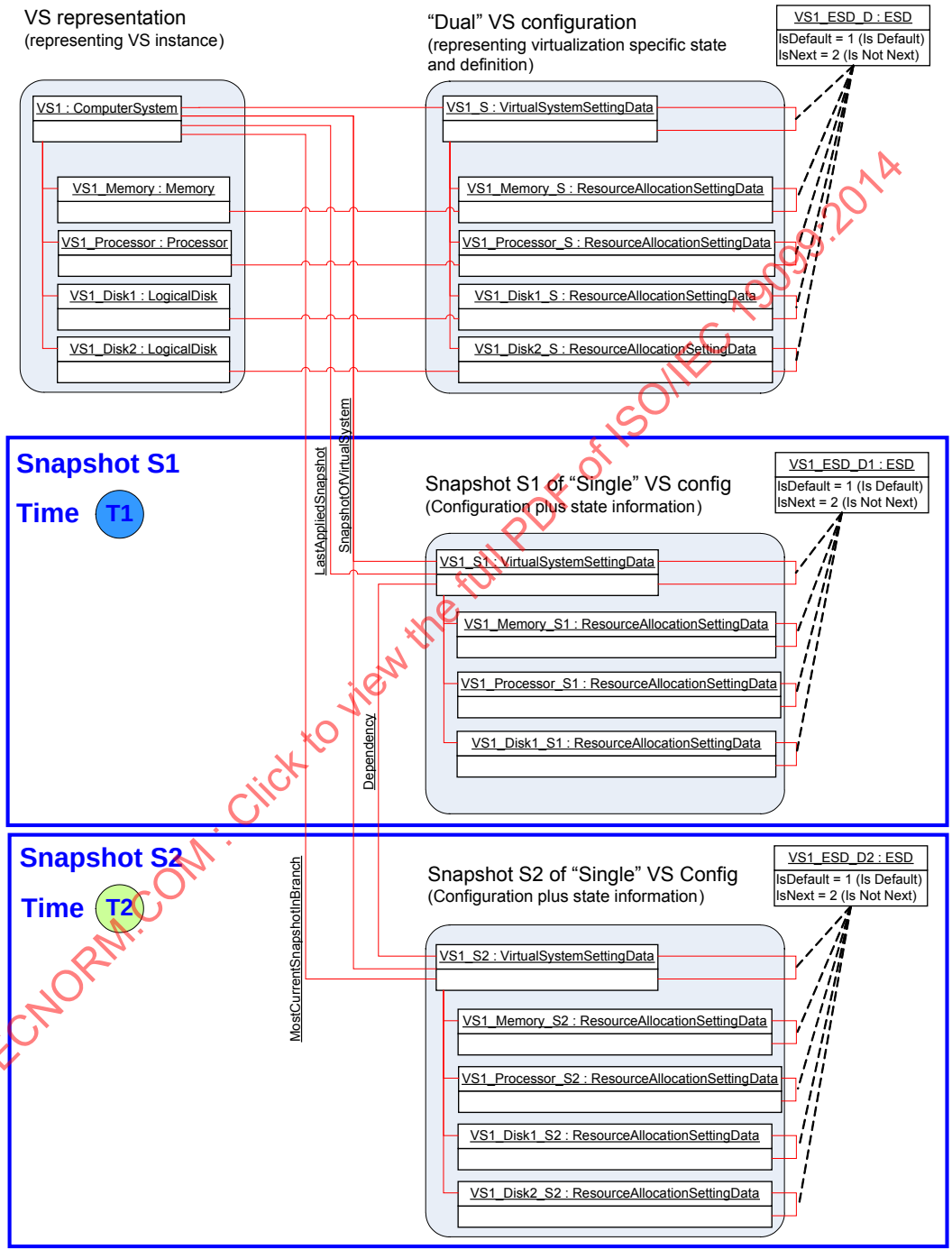

Figure 12 – System Virtualization Profile: Snapshot example

General assumption: The client knows the reference to an instance of the CIM_VirtualSystemSnapshotService class that represents the virtual system snapshot of a host system (see 6.4.2.6).

6.4.4.1 Locate virtual system snapshot service

Assumption: The client knows a reference to an instance of the CIM_System class that represents a host system that is a central instance of the profile described in this clause; see 6.4.2.4.

- The client invokes the intrinsic AssociatorNames( ) CIM operation as follows:
  - The value of the ObjectName parameter is set to refer to the instance of the CIM_System class.
  - The value of the AssocClass parameter is set to "CIM_HostedService".
  - The value of the ResultClass parameter is set to "CIM_VirtualSystemSnapshotService".

Result: The client knows a reference to the instance of the CIM_VirtualSystemSnapshotService class that represents the virtual system snapshot service serving the host system. If the operation is successful, the size of the result set is 1.

In the example shown in Figure 9, one instance of the CIM_VirtualSystemSnapshotService class serves the host system; it is tagged VSSS_1.

6.4.4.2 Determine capabilities of a virtual system snapshot service

Assumption: The client knows a reference to an instance of the CIM_VirtualSystemSnapshotService class that represents the virtual system snapshot service serving a host system (see 6.4.4.1).

1) The client invokes the intrinsic Associators( ) CIM operation as follows:
   - The value of the ObjectName parameter is set to refer to the instance of the CIM_VirtualSystemSnapshotService class.
   - The value of the AssocClass parameter is set to "CIM_ElementCapabilities".
   - The value of the ResultClass parameter is set to "CIM_VirtualSystemSnapshotServiceCapabilities".

   The result is a list of instances of the CIM_VirtualSystemSnapshotServiceCapabilities class. If the operation is successful, the size of the result set is 1.
2) The client analyzes the instance of the CIM_VirtualSystemSnapshotServiceCapabilities class.
   - The SynchronousMethodsSupported[ ] array property lists identifiers of methods of the CIM_VirtualSystemSnapshotServiceCapabilities class that are implemented with synchronous method execution only.
   - The AsynchronousMethodsSupported[ ] array property lists identifiers of methods of the CIM_VirtualSystemSnapshotServiceCapabilities class that are implemented with synchronous and asynchronous method execution.
   - The SnapshotTypesSupported[ ] array property lists identifiers designating snapshot types that are supported by the implementation.

Result: The client knows the virtual-system-snapshot-related capabilities of the host system in terms of properties of the CIM_VirtualSystemSnapshotServiceCapabilities class.

In the example shown in Figure 9, one instance of the CIM_VirtualSystemSnapshotServiceCapabilities class is associated with the host system; it is tagged VSSSC_1.

6.4.4.3 Create snapshot

Assumption: The client knows a reference to an instance of the CIM_ComputerSystem class that represents a virtual system hosted by a host system (see 6.4.2.16). The virtual system is active.

1) The client invokes the CreateSnapshot( ) method on the virtual system snapshot service, as follows:
   - The value of the AffectedSystem parameter is set to refer to the instance of the CIM_ComputerSystem class that represents the virtual system.
   - The value of the SnapshotType parameter is set to 2 (Full Snapshot).
2) The implementation executes the CreateSnapshot( ) method.

   The value returned in the ResultingSnapshot parameter refers to an instance of the CIM_VirtualSystemSettingData class that represents the new snapshot.

Result: The client knows a reference to the instance of the CIM_VirtualSystemSettingData class that represents the created virtual system snapshot.

In the example shown in Figure 12, two instances of the CIM_VirtualSystemSettingData class represent virtual system snapshots S1 and S2 taken at times T1 and T2. Although the situation captured in Figure 12 shows the situation at T3, a snapshot taken at T3 would look identical to S2 (because the current system at time T3 is unchanged with respect to S2).

6.4.4.4 Locate snapshots of a virtual system

Assumption: The client knows a reference to an instance of the CIM_ComputerSystem class that represents a virtual system (see 6.4.2.16).

- The client invokes the intrinsic Associators( ) CIM operation for the list of snapshots, as follows:
  - The value of the ObjectName parameter is set to refer to the instance of the CIM_ComputerSystem class.
  - The value of the AssocClass parameter is set to "CIM_SnapshotOfVirtualSystem".
  - The value of the ResultClass parameter is set to "CIM_VirtualSystemSettingData".

  The result is a list of instances of the CIM_VirtualSystemSettingData class.

Result: The client knows a set of instances of the CIM_VirtualSystemSettingData class, each representing a virtual system snapshot taken from the virtual system.

In the example shown in Figure 12, the instances tagged VS_S1 and VS1_S2 of the CIM_VirtualSystemSettingData class represent snapshots S1 and S2.

6.4.4.5 Locate the source virtual system of a snapshot

Assumption: The client knows the reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot.

- The client invokes the intrinsic AssociatorNames( ) CIM operation for the source virtual system as follows:
  - The value of the ObjectName parameter is set to refer to the instance of the CIM_VirtualSystemSettingData class.
  - The value of the AssocClass parameter is set to "CIM_ElementSettingData".
  - The value of the ResultClass parameter is set to "CIM_ComputerSystem".

The result is a list of references to instances of the CIM_ComputerSystem class. The size of the list is 1.

Result: The client knows a reference to an instance of the CIM_ComputerSystem class that represents the virtual system that was the source for the snapshot.

NOTE At this time the present configuration of the virtual system may be completely different from the configuration that was captured in the snapshot.

In the example shown in Figure 12, the instance of class CIM_ComputerSystem tagged VS1 is the source of snapshots S1 and S2, represented by instances of the CIM_VirtualSystemSettingData class tagged VS_S1 and VS_S2.

6.4.4.6 Locate the most current snapshot in a branch of snapshots

Assumption: The client knows an instance of the CIM_ComputerSystem class that represents a virtual system (see 6.4.2.16).

- The client invokes the intrinsic Associators( ) CIM operation for the most current snapshot in the current branch of virtual snapshots, as follows:
  - The value of the ObjectName parameter is set to refer to the instance of the CIM_ComputerSystem class.
  - The value of the AssocClass parameter is set to "CIM_MostCurrentSnapshotInBranch".
  - The value of the ResultClass parameter is set to "CIM_VirtualSystemSettingData".

  The result is a list of instances of the CIM_VirtualSystemSettingData class. The size of the list is 1.

Result: The client knows an instance of the CIM_VirtualSystemSettingData class that represents the virtual system snapshot that is the most current snapshot in the current branch of snapshots.

In the example shown in Figure 12, the instance of the CIM_VirtualSystemSettingData class that is tagged VS1_2 represents the most current snapshot in the current branch of snapshots. This is the case because that snapshot was applied most recently to the virtual system and no other snapshot was applied to or created from the virtual system since then.

6.4.4.7 Locate dependent snapshots

Assumption: The client knows a reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot (see 6.4.4.4).

- The client invokes the intrinsic AssociatorNames( ) CIM operation for the list of dependent snapshots as follows:
  - The value of the ObjectName parameter is set to refer to the instance of the CIM_VirtualSystemSettingData class.
  - The value of the AssocClass parameter is set to "CIM_Dependency".
  - The value of the ResultClass parameter is set to "CIM_VirtualSystemSettingData".
  - The value of the Role parameter is set to "Antecedent".
  - The value of the ResultRole parameter is set to "Dependent".

  The result is a list of references to instances of the CIM_VirtualSystemSettingData class.

Result: The client knows a set of instances of the CIM_VirtualSystemSettingData class that represent virtual system snapshots that depend on the input virtual system snapshot. The set may be empty, indicating that no dependent snapshots exist.

In the example shown in Figure 12, the instance tagged VS_S2 represents snapshot S2, which is dependent on snapshot S1, which is represented by the instance tagged VS_S1.

6.4.4.8 Locate parent snapshot

Assumption: The client knows a reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot (see 6.4.4.4).

- The client invokes the intrinsic AssociatorNames( ) CIM operation for the parent snapshot as follows:
  - The value of the ObjectName parameter is set to refer to the instance of the CIM_VirtualSystemSettingData class that represents the virtual system snapshot.
  - The value of the AssocClass parameter is set to "CIM_Dependency".
  - The value of the ResultClass parameter is set to "CIM_VirtualSystemSettingData".
  - The value of the Role parameter is set to "Dependent".
  - The value of the ResultRole parameter is set to "Antecedent".

  The result is a list of references to instances of the CIM_VirtualSystemSettingData class that represent virtual system snapshots. The list has a size of 1 or 0.

Result: The client knows the instance of the CIM_VirtualSystemSettingData class that represents the parent virtual system snapshot of the input virtual system snapshot. The set may be empty, indicating that no parent snapshots exist.

In the example shown in Figure 12, the instance tagged VS_S1 represents snapshot S1, which is the parent of snapshot S2, which is represented by the instance tagged VS_S2.

6.4.4.9 Apply snapshot

Assumption: The client knows a reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot (see 6.4.4.3 or 6.4.4.4). The client knows a reference to the instance of the CIM_ComputerSystem class that represents the virtual system that was the source for the snapshot (see 6.4.4.5). The virtual system is active.

1) The client invokes the ApplySnapshot( ) method on the virtual system snapshot service. The value of the Snapshot parameter is set to refer to the instance of the CIM_VirtualSystemSettingData class that represents the snapshot.
2) The snapshot is applied into the active virtual system as follows:
   a) The virtual system is deactivated. This implies a disruptive termination of the software that may be active in the instance of the virtual system.
   b) The virtual system is reconfigured according to the virtual system snapshot. For a disk snapshot, this applies to the disk resources only.
   c) If the applied snapshot is a full snapshot, all stateful resources like memory and disk are restored to the situation that was captured in the snapshot. If the applied snapshot is a disk snapshot, only disk resources are restored.
   d) The virtual system is activated. If the applied snapshot is a full snapshot, the virtual system starts from the situation that was captured by the full snapshot. If the applied snapshot was a disk snapshot, a normal virtual system activation occurs.

Result: The virtual system is restored to the situation that was in place when the snapshot was taken.

In the example shown in Figure 12, the situation is depicted at time T3, immediately after the activation of snapshot S2 within virtual system VS1.

6.4.4.10 Destroy snapshot

Assumption: The client knows the reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot (see 6.4.2.16).

1) The client invokes the DestroySnapshot( ) method on the virtual system management service. The value of the Snapshot parameter is set to refer to the instance of the CIM_VirtualSystemSettingData class that represents the snapshot.
2) The snapshot is removed from the implementation.

Result: The snapshot no longer exists within the implementation.

6.5 CIM elements

Table 62 lists CIM elements that are defined or specialized for the profile described in this clause. Each CIM element shall be implemented as described in Table 62. The CIM Schema descriptions for any referenced element and its sub-elements apply.

Subclauses 6.2 ("Implementation") and 6.3 ("Methods") may impose additional requirements on these elements.

Table 62 – CIM Elements: System Virtualization Profile

Element name	Requirement	Description
CIM_AffectedJobElement	Conditional	See 6.5.1.
CIM_ConcreteJob	Conditional	See 6.5.2.
CIM_Dependency	Conditional	See 6.5.3.
CIM_ElementCapabilities (Host system)	Mandatory	See 6.5.4.
CIM_ElementCapabilities (Virtual system management service)	Mandatory	See 6.5.5.
CIM_ElementCapabilities (Virtual system snapshot service)	Conditional	See 6.5.6.
CIM_ElementCapabilities (Snapshots of virtual systems)	Conditional	See 6.5.7.
CIM_ElementConformsToProfile	Mandatory	See 6.5.8.
CIM_HostedDependency	Mandatory	See 6.5.9.
CIM_HostedService (Virtual system management service)	Conditional	See 6.5.10.
CIM_HostedService (Virtual system snapshot service)	Conditional	See 6.5.11.
CIM_LastAppliedSnapshot	Conditional	See 6.5.12.
CIM_MostCurrentSnapshotInBranch	Conditional	See 6.5.13.
CIM_ReferencedProfile	Conditional	See 6.5.14.
CIM_RegisteredProfile	Mandatory	See 6.5.15.
CIM_ServiceAffectsElement (Virtual system management service)	Conditional	See 6.5.16.
CIM_ServiceAffectsElement (Virtual system snapshot service)	Conditional	See 6.5.17.
CIM_SnapshotOfVirtualSystem	Conditional	See 6.5.18.
CIM_System	Mandatory	See 6.5.19.
CIM_VirtualSystemManagementCapabilities	Mandatory	See 6.5.20.

Element name	Requirement	Description
CIM_VirtualSystemManagementService	Conditional	See 6.5.21.
CIM_VirtualSystemSettingData (Input)	Conditional	See 6.5.22.
CIM_VirtualSystemSettingData (Snapshot)	Conditional	See 6.5.23.
CIM_VirtualSystemSnapshotCapabilities	Conditional	See 6.5.24.
CIM_VirtualSystemSnapshotService	Optional	See 6.5.25.
CIM_VirtualSystemSnapshotServiceCapabilities	Conditional	See 6.5.26.

6.5.1 CIM_AffectedJobElement

The implementation of the CIM_AffectedJobElement association is conditional.

Condition: A non-NULL value for at least one element of the AsynchronousMethodsSupported[ ] array property of the CIM_VirtualSystemManagementCapabilities class is implemented.

If the CIM_AffectedJobElement association is implemented, the provisions in this subclause apply.

An implementation shall use the CIM_AffectedJobElement association to associate an instance of the CIM_ConcreteJob class that represents an asynchronous task and an instance of the CIM_ComputerSystem class that represents a virtual system that is affected by its execution.

Table 63 contains the requirements for elements of this association.

Table 63 – Association: CIM_AffectedJobElement

Elements	Requirement	Notes
AffectedElement	Mandatory	**Key:** See 6.3.1.2. **Cardinality:** *
AffectingElement	Mandatory	**Key:** See 6.3.1.2. **Cardinality:** 1
ElementEffects[ ]	Mandatory	See 6.3.1.2.

6.5.2 CIM_ConcreteJob

The implementation of the CIM_ConcreteJob class is conditional.

Condition: A non-NULL value for at least one element of the AsynchronousMethodsSupported[ ] array property of the CIM_VirtualSystemManagementCapabilities class is implemented.

If the CIM_ConcreteJob class is implemented, the provisions in this subclause apply.

An implementation shall use an instance of the CIM_ConcreteJob class to represent an asynchronous task.

Table 64 contains requirements for elements of this class.

Table 64 – Class: CIM_ConcreteJob

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
JobState	Mandatory	See 6.3.1.2.
TimeOfLastStateChange	Mandatory	See 6.3.1.2.

6.5.3 CIM_Dependency

The implementation of the CIM_Dependency association is conditional.

Condition: Virtual system snapshots are implemented; see 6.2.7.1.2.

If the CIM_Dependency association class is implemented, the provisions in this subclause apply.

An implementation shall use an instance of the CIM_Dependency association to associate an instance of the CIM_VirtualSystemSettingData class that represents a parent snapshot and an instance of the CIM_VirtualSystemSettingData class that represents a dependent snapshot.

Table 65 contains requirements for elements of this class.

Table 65 – Class: CIM_Dependency Class

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Reference to an instance of the CIM_VirtualSystemSettingData class that represents a parent snapshot **Cardinality:** 0..1
Dependent	Mandatory	**Key:** Reference to an instance of the CIM_VirtualSystemSettingData class that represents a dependent snapshot **Cardinality:** 0..1

6.5.4 CIM_ElementCapabilities (host system)

An implementation shall use an instance of the CIM_ElementCapabilities association to associate an instance of the CIM_System class that represents a host system with an instance of the CIM_VirtualSystemManagementCapabilities class that describes the virtual system management capabilities of the host system.

Table 66 contains requirements for elements of this association.

Table 66 – Association: CIM_ElementCapabilities (host system)

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Reference to instance of the CIM_System class that represents a host system **Cardinality:** 1
Capabilities	Mandatory	**Key:** Reference to an instance of the CIM_VirtualSystemManagementCapabilities class that describes the capabilities of a host system **Cardinality:** 1

6.5.5 CIM_ElementCapabilities (virtual system management service)

The implementation of the CIM_ElementCapabilities association for the virtual system management service is conditional.

Condition: Any of the following is implemented:

- Virtual system definition and destruction (see 6.2.4.6.1)
- Virtual resource addition and removal (see 6.2.4.6.2)
- Virtual system and resource modification (see 6.2.4.6.3)

If the CIM_ElementCapabilities association is implemented for the virtual system management service, the provisions in this subclause apply.

An implementation shall use an instance of the CIM_ElementCapabilities association to associate an instance of the CIM_VirtualSystemManagementService class that represents a virtual system management service with an instance of the CIM_VirtualSystemManagementCapabilities that describes the capabilities of the virtual system management service.

Table 67 contains requirements for elements of this association.

Table 67 – Association: CIM_ElementCapabilities (virtual system management)

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Reference to instance of the CIM_VirtualSystemManagementService class **Cardinality:** 0..1
Capabilities	Mandatory	**Key:** Reference to an instance of the CIM_VirtualSystemManagementCapabilities class **Cardinality:** 1

6.5.6 CIM_ElementCapabilities (virtual system snapshot service)

The implementation of the CIM_ElementCapabilities association for the virtual system snapshot service is conditional.

Condition: Virtual system snapshots are implemented; see 6.2.7.1.2.

If the CIM_ElementCapabilities association is implemented for the virtual system snapshot service, the provisions in this subclause apply.

An implementation shall use an instance of the CIM_ElementCapabilities association to associate an instance of the CIM_VirtualSystemSnapshotService class that represents a virtual system snapshot service with an instance of the CIM_VirtualSystemSnapshotServiceCapabilities class that describes the capabilities of the virtual system snapshot service.

Table 68 contains requirements for elements of this association.

Table 68 – Association: CIM_ElementCapabilities (snapshot service)

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Reference to an instance of the CIM_VirtualSystemSnapshotService class that represents a virtual system snapshot service **Cardinality:** 1
Capabilities	Mandatory	**Key:** Reference to the instance of the CIM_VirtualSystemSnapshotServiceCapabilities class that represents the capabilities of the virtual system snapshot service **Cardinality:** 1

6.5.7 CIM_ElementCapabilities (snapshots of virtual systems)

The implementation of the CIM_ElementCapabilities association for the virtual systems snapshots is conditional.

Condition: Virtual system snapshots are implemented; see 6.2.7.1.2.

If the CIM_ElementCapabilities association is implemented for virtual systems snapshots, the provisions in this subclause apply.

The implementation shall use an instance of the CIM_ElementCapabilities association to associate instances of the CIM_VirtualSystemSnapshotCapabilities class with those instances of the CIM_ComputerSystem class that represent a virtual system to which the capabilities apply.

Table 69 contains requirements for elements of this association.

Table 69 – Association: CIM_ElementCapabilities (snapshots of virtual systems)

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Reference to an instance of the CIM_ComputerSystem class that represents a virtual system **Cardinality:** *
Capabilities	Mandatory	**Key:** Reference to the instance of the CIM_VirtualSystemSnapshotCapabilities class that describes the current applicability of snapshot related services to the virtual system **Cardinality:** 1

6.5.8 CIM_ElementConformsToProfile

An implementation shall use an instance of the CIM_ElementConformsToProfile association to associate an instance of the CIM_RegisteredProfile class that represents an implementation of the profile described in this clause with instances of the CIM_System class that represent a host system that is a central and scoping instance of the profile described in this clause.

Table 70 contains requirements for elements of this association.

Table 70 – Association: CIM_ElementConformsToProfile

Elements	Requirement	Notes
ConformantStandard	Mandatory	**Key:** Reference to an instance of the CIM_RegisteredProfile class that represents an implementation of the profile described in this clause **Cardinality:** 1
ManagedElement	Mandatory	**Key:** Reference to an instance of the CIM_ System class that represents a host system **Cardinality:** *

6.5.9 CIM_HostedDependency

An implementation shall use an instance of the CIM_HostedDependency association to associate an instance of the CIM_System class that represents a host system with each instance of the CIM_ComputerSystem class that represents a virtual system hosted by the host system.

Table 71 contains requirements for elements of this association.

Table 71 – Association: CIM_HostedDependency

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Reference to an instance of the CIM_System class that represents a host system **Cardinality:** 1
Dependent	Mandatory	**Key:** Reference to an instance of the CIM_ComputerSystem class that represents a virtual system **Cardinality:** *

6.5.10 CIM_HostedService (virtual system management service)

The implementation of the CIM_HostedService association for the virtual system management service is conditional:

Condition: Any of the following is implemented:

- Virtual system definition and destruction (see 6.2.4.6.1)
- Vvirtual resource addition and removal (see 6.2.4.6.2)
- Virtual system and resource modification (see 6.2.4.6.3)

If the CIM_HostedService association is implemented for the virtual system management service, the provisions in this subclause apply.

The implementation shall use an instance of the CIM_HostedService association to associate an instance of the CIM_System class that represents a host system and the instance of the CIM_VirtualSystem-ManagementService class that represents the virtual system management service that is hosted by a host system.

Table 72 contains requirements for elements of this association.

Table 72 – Association: CIM_HostedService (virtual system management service)

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Reference to an instance of the CIM_System class that represents a host system **Cardinality:** 1
Dependent	Mandatory	**Key:** Reference to an instance of the CIM_VirtualSystemManagementService class that represents a virtual system management service **Cardinality:** 0..1

6.5.11 CIM_HostedService (virtual system snapshot service)

The implementation of the CIM_HostedService association is conditional.

Condition: Virtual system snapshots are implemented; see 6.2.7.1.2.

If the CIM_HostedService association is implemented for the virtual system snapshot service, the provisions in this subclause apply.

The implementation shall use an instance of the CIM_HostedService association to associate an instance of the CIM_ComputerSystem class that represents a host system and the instance of the CIM_VirtualSystemSnapshotService class that represents the virtual system snapshot service.

Table 73 contains requirements for elements of this association.

Table 73 – Association: CIM_HostedService (virtual system snapshot service)

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Reference to an instance of the CIM_System class that represents a host system **Cardinality:** 1
Dependent	Mandatory	**Key:** Reference to an instance of the CIM_VirtualSystemSnapshotService class that represents a virtual system snapshot service **Cardinality:** 0..1

6.5.12 CIM_LastAppliedSnapshot

The implementation of the CIM_LastAppliedSnapshot association is conditional.

Condition: Virtual system snapshots are implemented; see 6.2.7.1.2.

If the CIM_LastAppliedSnapshot association is implemented, the provisions in this subclause apply.

An implementation shall use an instance of the CIM_LastAppliedSnapshot association to associate an instance of the CIM_ComputerSystem class that represents a virtual system and the instance of the CIM_VirtualSystemSettingData class that represents the virtual system snapshot that was last applied to the virtual system.

Table 74 contains requirements for elements of this association.

Table 74 – Association: CIM_LastAppliedSnapshot

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot **Cardinality:** 0..1
Dependent	Mandatory	**Key:** Reference to the instance of the CIM_ComputerSystem class that represents the virtual system **Cardinality:** 0..1

6.5.13 CIM_MostCurrentSnapshotInBranch

The implementation of the CIM_MostCurrentSnapshotInBranch association is conditional.

Condition: Virtual system snapshots are implemented; see 6.2.7.1.2.

If the CIM_MostCurrentSnapshotInBranch association is implemented, the provisions in this subclause apply.

An implementation shall use an instance of the CIM_MostCurrentSnapshotInBranch association to associate an instance of the CIM_ComputerSystem class that represents a virtual system and the instance of the CIM_VirtualSystemSettingData class that represents the most current snapshot in a branch of virtual system snapshots. The most current snapshot in a branch of snapshots related to an instance of a virtual system is the younger of the following snapshots:

- the snapshot that was most recently captured from the virtual system instance
- the snapshot that was last applied to the instance

Table 75 contains requirements for elements of this association.

Table 75 – Association: CIM_MostCurrentSnapshotInBranch

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Reference to the instance of the CIM_ComputerSystem class that represents the virtual system **Cardinality:** 0..1
Dependent	Mandatory	**Key:** Reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot **Cardinality:** 0..1

6.5.14 CIM_ReferencedProfile

The implementation of the CIM_ReferencedProfile association is conditional.

Condition: Resource virtualization profiles such as the Generic Device Resource Virtualization Profile described in clause 13 are implemented as scoped profiles.

If the CIM_ReferencedProfile association is implemented, the provisions in this subclause apply.

An implementation shall use an instance of the CIM_ReferencedProfile association to associate an instance of the CIM_RegisteredProfile class that represents an implementation of the profile described in this clause and any instance of the CIM_RegisteredProfile class that represents an implementation of a resource allocation DMTF management profile that describes virtual resource allocation that is implemented by the implementation.

Table 76 contains requirements for elements of this association.

Table 76 – Association: CIM_ReferencedProfile

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Reference to an instance of the CIM_RegisteredProfile that represents an implementation of the profile described in this clause **Cardinality:** 1
Dependent	Mandatory	**Key:** Reference to an instance of the CIM_RegisteredProfile class that represents an implementation of a resource allocation profile **Cardinality:** *

6.5.15 CIM_RegisteredProfile

An implementation shall use an instance of the CIM_RegisteredProfile class to represent an implementation of the profile described in this clause.

Table 77 contains requirements for elements of this class.

Table 77 – Class: CIM_RegisteredProfile

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
RegisteredOrganization	Mandatory	Shall be set to "DMTF".
RegisteredName	Mandatory	Shall be set to "System Virtualization".
RegisteredVersion	Mandatory	Shall be set to the version of the profile described in this clause ("1.0.0").

6.5.16 CIM_ServiceAffectsElement (virtual system management service)

The implementation of the CIM_ServiceAffectsElement association for the virtual system management service is conditional.

Condition: Any of the following is implemented:

- Virtual system definition and destruction (see 6.2.4.6.1)
- Virtual resource addition and removal (see 6.2.4.6.2)
- Virtual system and resource modification (see 6.2.4.6.3)

If the CIM_ServiceAffectsElement association is implemented for the virtual system management service, the provisions in this subclause apply.

The implementation shall use an instance of the CIM_ServiceAffectsElement association to associate an instance of the CIM_VirtualSystemManagementService class that represents a virtual system management service and any instance of the CIM_ComputerSystem class that represents a virtual system that is managed by that virtual system management service.

Table 78 contains requirements for elements of this association.

Table 78 – Association: CIM_ServiceAffectsElement (virtual system management service)

Elements	Requirement	Notes
AffectedElement	Mandatory	**Key:** Reference to instance of the CIM_ComputerSystem class that represents a managed virtual system **Cardinality:** *
AffectingElement	Mandatory	**Key:** Reference to an instance of the CIM_VirtualSystemManagementService class that represents a virtual system management service **Cardinality:** 0..1

6.5.17 CIM_ServiceAffectsElement (virtual system snapshot service)

The implementation of the CIM_ServiceAffectsElement association is conditional.

Condition: Virtual system snapshots are implemented; see 6.2.7.1.2.

If the CIM_ServiceAffectsElement association is implemented for the virtual system snapshot service, the provisions in this subclause apply.

The implementation shall use an instance of the CIM_ServiceAffectsElement association to associate an instance of the CIM_VirtualSystemSnapshotService class that represents a virtual system management service with the following instances:

- any instance of the CIM_ComputerSystem class that represents a virtual system that is managed by that virtual system management service
- any instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot

Table 79 contains requirements for elements of this association.

Table 79 – Association: CIM_ServiceAffectsElement

Elements	Requirement	Notes
AffectedElement	Mandatory	**Key:** Reference to instance of the CIM_ComputerSystem class that represents a virtual system or the CIM_VirtualSystemSettingData class that represents a managed snapshot **Cardinality:** *
AffectingElement	Mandatory	**Key:** Reference to an instance of the CIM_VirtualSystemManagementService class that represents a virtual system snapshot service **Cardinality:** 0..1

6.5.18 CIM_SnapshotOfVirtualSystem

The implementation of the CIM_SnapshotOfVirtualSystem association is conditional.

Condition: Virtual system snapshots are implemented; see 6.2.7.1.2

If the CIM_SnapshotOfVirtualSystem association is implemented, the provisions in this subclause apply.

An implementation shall use an instance of the CIM_SnapshotOfVirtualSystem association to associate an the instance of the CIM_ComputerSystem class that represents the virtual system that was the source for the virtual system snapshot and the instance of the CIM_VirtualSystemSettingData class that represents a snapshot of the virtual system

Table 80 contains requirements for elements of this association.

Table 80 – Association: CIM_SnapshotOfVirtualSystem

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Reference to the instance of the CIM_ComputerSystem class that represents the source virtual system **Cardinality:** 0..1
Dependent	Mandatory	**Key:** Reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot **Cardinality:** *

6.5.19 CIM_System

An implementation shall use an instance of a concrete subclass of the CIM_System class to represent a host system.

Table 81 contains requirements for elements of this class.

Table 81 – Class: CIM_VirtualSystemManagementCapabilities

Elements	Requirement	Notes
CreationClassName	Mandatory	**Key**
Name	Mandatory	**Key**

6.5.20 CIM_VirtualSystemManagementCapabilities

An implementation shall use an instance of the CIM_VirtualSystemManagementCapabilities class to represent the virtual system management capabilities of a host system.

Table 82 contains requirements for elements of this class.

Table 82 – Class: CIM_VirtualSystemManagementCapabilities

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
VirtualSystemTypesSupported[ ]	Optional	See 6.2.4.2.
SynchronousMethodsSupported[ ]	Optional	See 6.2.4.3.
AsynchronousMethodsSupported[ ]	Optional	See 6.2.4.4.
IndicationsSupported[ ]	Optional	See 6.2.4.5.

6.5.21 CIM_VirtualSystemManagementService

The implementation of the CIM_VirtualSystemManagementService class is conditional.

Condition: Any of the following is implemented:

- Virtual system definition and destruction (see 6.2.4.6.1)
- Virtual resource addition and removal (see 6.2.4.6.2)
- Virtual system and resource modification (see 6.2.4.6.3)

If the CIM_VirtualSystemManagementService class is implemented, the provisions in this subclause apply.

An implementation shall use an instance of the CIM_VirtualSystemManagementService class to represent the virtual system management service provided by one host system.

Table 83 contains requirements for elements of this class.

Table 83 – Class: CIM_VirtualSystemManagementService

Elements	Requirement	Notes
CreationClassName	Mandatory	**Key**
Name	Mandatory	**Key**
SystemCreationClassName	Mandatory	**Key**
SystemName	Mandatory	**Key**

Elements	Requirement	Notes
AddResourceSettings( )	Conditional	See 6.3.2.3.
DefineSystem( )	Conditional	See 6.3.2.1.
DestroySystem( )	Conditional	See 6.3.2.2.
ModifyResourceSettings( )	Conditional	See 6.3.2.4.
ModifySystemSettings( )	Conditional	See 6.3.2.5.
RemoveResourceSettings( )	Conditional	See 6.3.2.6.

6.5.22 CIM_VirtualSystemSettingData (input)

The implementation of the CIM_VirtualSystemSettingData class for input is conditional.

Condition: Any of the following is implemented:

- Virtual system definition and destruction (see 6.2.4.6.1)
- Virtual resource addition and removal (see 6.2.4.6.2)
- Virtual system and resource modification (see 6.2.4.6.3)

If the CIM_VirtualSystemSettingData class is implemented for input, the provisions in this subclause apply.

An instance of the CIM_VirtualSystemSettingData class shall be used to represent input data for a virtual system's definitions and modifications.

Table 84 contains requirements for elements of this class.

Table 84 – Class: CIM_VirtualSystemSettingData (input)

Elements	Requirement	Notes
InstanceID	Mandatory	**Key** (Input): See 6.2.5.1.
ElementName	Optional	See 6.2.5.2.
VirtualSystemIdentity	Optional	See 6.2.5.3.
VirtualSystemType	Optional	See 6.2.5.4.

6.5.23 CIM_VirtualSystemSettingData (snapshot)

The implementation of the CIM_VirtualSystemSettingData class for the representation of snapshots of virtual systems is conditional.

Condition: Virtual system snapshots are implemented; see 6.2.7.1.2.

If the CIM_VirtualSystemSettingData class is implemented for the representation of snapshots, the provisions in this subclause apply.

An instance of the CIM_VirtualSystemSettingData class shall be used to represent snapshots of virtual systems.

Table 85 contains requirements for elements of this class.

Table 85 – Class: CIM_VirtualSystemSettingData (Snapshot)

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
Caption	Optional	See CIM Schema.
Description	Optional	See CIM Schema.
ElementName	Optional	See CIM Schema.
VirtualSystemIdentifier	Optional	See CIM Schema.
VirtualSystemType	Optional	See CIM Schema.
Notes	Optional	See CIM Schema.
CreationTime	Mandatory	The value shall reflect the creation time of the snapshot.
ConfigurationID	Optional	See CIM Schema.
ConfigurationDataRoot	Optional	See CIM Schema.
ConfigurationFile	Mandatory	This element shall have a value of NULL.
SnapshotDataRoot	Mandatory	This element shall have a value of NULL.
SuspendDataRoot	Optional	See CIM Schema.
SwapFileDataRoot	Mandatory	This element shall have a value of NULL.
LogDataRoot	Optional	See CIM Schema.
AutomaticStartupAction	Mandatory	This element shall have a value of NULL.
AutomaticStartupActionDelay	Mandatory	This element shall have a value of NULL.
AutomaticStartupActionSequenceNumber	Mandatory	This element shall have a value of NULL.
AutomaticShutdownAction	Mandatory	This element shall have a value of NULL.
AutomaticRecoveryAction	Mandatory	This element shall have a value of NULL.
RecoveryFile	Mandatory	This element shall have a value of NULL.
NOTE: Elements marked as mandatory but with a required value of NULL shall in effect not be implemented. Respective information applies to the virtual system as a whole, not just to a particular snapshot, and is covered by the instance of the CIM_VirtualSystemSettingData class in the "State" and the "Defined" virtual system configuration.		

6.5.24 CIM_VirtualSystemSnapshotCapabilities

The implementation of the CIM_VirtualSystemSnapshotCapabilities class is optional.

If the CIM_VirtualSystemSnapshotCapabilities class is implemented, the provisions in this subclause apply.

The implementation of the optional CIM_VirtualSystemSnapshotCapabilities class is specified only if virtual system snapshots are implemented; see 6.2.7.1.2.

An instance of the CIM_VirtualSystemSnapshotCapabilities class may be used to represent the current applicability of snapshot-related services to one virtual system.

Table 86 contains requirements for elements of this class.

Table 86 – Class: CIM_VirtualSystemSnapshotCapabilities

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
SnapshotTypesEnabled[ ]	Mandatory	See 6.2.7.5.1.
GuestOSNotificationEnabled[ ]	Optional	See 6.2.7.5.2.

6.5.25 CIM_VirtualSystemSnapshotService

The implementation of the CIM_VirtualSystemSnapshotService class is optional.

If the CIM_VirtualSystemSnapshotService class is implemented, the provisions in this subclause apply.

If the CIM_VirtualSystemSnapshotService class is implemented, this indicates the presence of the support of virtual system snapshots (see 6.2.7.1.2).

An instance of the CIM_VirtualSystemSnapshotService class shall be used to represent the virtual system snapshot service available at a host system.

Table 87 contains requirements for elements of this class.

Table 87 – Class: CIM_VirtualSystemSnapshotService

Elements	Requirement	Notes
CreationClassName	Mandatory	**Key**
Name	Mandatory	**Key**
SystemCreationClassName	Mandatory	**Key**
SystemName	Mandatory	**Key**
CreateSnapshot( )	Conditional	See 6.3.3.1.
DestroySnapshot( )	Conditional	See 6.3.3.2.
ApplySnapshot( )	Conditional	See 6.3.3.3.

6.5.26 CIM_VirtualSystemSnapshotServiceCapabilities

The implementation of the CIM_VirtualSystemSnapshotServiceCapabilities class is conditional.

Condition: Virtual system snapshots are implemented; see 6.2.7.1.2.

If the CIM_VirtualSystemSnapshotServiceCapabilities class is implemented, the provisions in this subclause apply.

An instance of the CIM_VirtualSystemSnapshotServiceCapabilities class shall be used to represent the capabilities of a virtual system snapshot service.

Table 88 contains requirements for elements of this class.

Table 88 – Class: CIM_VirtualSystemSnapshotServiceCapabilities

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
SynchronousMethodsSupported[ ]	Conditional	See 6.2.7.1.3.
AsynchronousMethodsSupported[ ]	Conditional	See 6.2.7.1.3.
SnapshotTypesSupported[ ]	Mandatory	See 6.2.7.1.3.

7 Allocation Capabilities Profile

Profile Name: Allocation Capabilities

Version: 1.0.0

Organization: DMTF

CIM schema version: 2.22

Central Class: CIM_AllocationCapabilities

Scoping Class: CIM_System

This abstract profile shall not be directly implemented; implementation shall be based on a profile specification for the capabilities of a resource pool or virtualization system for a specific class of resources, such as CPU and system memory. The scoping association paths between the central class and the scoping class shall be specified by the incorporating concrete profiles.

Table 89 identifies profiles on which the profile described in this clause has a dependency.

Table 89 – Related profiles for the Allocation Capabilities Profile

Profile name	Organization	Version	Requirement	Description
Resource Allocation	DMTF	1.0	Mandatory	Defines resource allocation setting data (see clause 5)

7.1 Description

The Allocation Capabilities Profile is an abstract profile that describes the use of the class CIM_AllocationCapabilities and the use of association CIM_SettingsDefineCapabilities to a set of CIM_ResourceAllocationSettingData instances to describe the default property values, supported property values, and range of property values for a resource allocation request.

Figure 13 represents the class schema for the Allocation Capabilities Profile. For simplicity, the prefix CIM_ has been removed from the names of the classes.

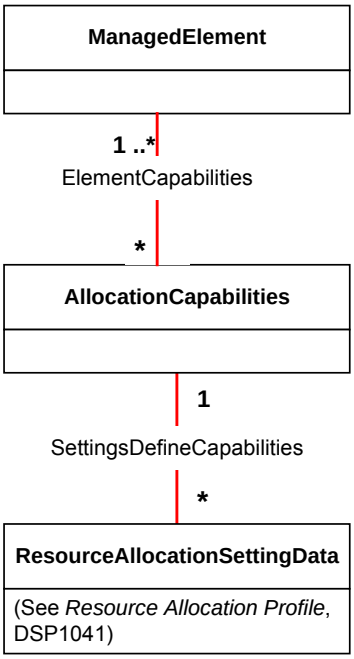

Figure 13 – Allocation Capabilities Profile: Class diagram

The CIM_ManagedElement class in Figure 13 represents a potential provider of resources such as a host system (CIM_ComputerSystem) or a resource pool (CIM_ResourcePool), or the CIM_ManagedElement class represents an instance of CIM_ResourceAllocationSettingData.

CIM_ElementCapabilities associates the CIM_AllocationCapabilities instance to a subclass of CIM_ManagedElement.

If a CIM_AllocationCapabilities instance associated using the CIM_ElementCapabilities association is used to define the allocation capabilities of the managed element, the set of CIM_ResourceAllocationSettingData instances together with properties of the CIM_AllocationCapabilities instance defines a supported set of default property values, supported property values, and range of supported property values required to form a valid allocation request for the resource type.

If a CIM_AllocationCapabilities instance is associated using the CIM_ElementCapabilities association to an instance of CIM_ResourceAllocationSettingData, the set of associated CIM_ResourceAllocationSettingData instances represents the supported changes or the range of valid changes for the properties in the CIM_ResourceAllocationSettingData instance.

CIM_SettingsDefineCapabilities associates instances of CIM_ResourceAllocationSettingData to a CIM_AllocationCapabilities instance and defines the type of capability the CIM_ResourceAllocationSettingData represents.

Note that the allocation capabilities do not reflect the current or dynamic state of any allocations. Rather they define valid allocation requests or valid settings modifications supported by the host system or resource pool without regard to the current availability of a host resource.

7.1.1 CIM_SettingsDefineCapabilities

The CIM_SettingsDefineCapabilities association indicates that the non-null, non-key set of properties of the component CIM_ResourceAllocationSettingData instance specifies some capabilities of the associated CIM_AllocationCapabilities instance. The interpretation of the set of properties in the

associated CIM_ResourceAllocationSettingData is governed by the CIM_SettingsDefineCapabilities properties: PropertyPolicy, ValueRole, and ValueRange.

7.1.1.1 PropertyPolicy

The CIM Schema description of this property applies.

PropertyPolicy defines whether or not the non-null, non-key properties of the associated CIM_ResourceAllocationSettingData instance are treated independently or as a correlated set.

The profile described in this clause assumes that the value of the PropertyPolicy property is 0 (“Independent”) if the ValueRange contains a value of 1 (“Minimums”), 2 (“Maximums”) or 3 (“Increments”). In these cases multiple instances of CIM_AllocationCapabilities with independent sets of capabilities are used to express correlated sets of capabilities.

PropertyPolicy can be set to 0 (“Independent”) or 1 (“Correlated”) if the property ValueRange is set to “Point” to express the independence or the dependence of the set of properties in the associated CIM_ResourceAllocationSettingData instance.

7.1.1.2 ValueRole

The CIM Schema description of this property applies.

The possible values for the ValueRole property are as follows:

- 0 (“Default“) indicates that property values of the component CIM_ResourceAllocationSettingData instance are the default values that are used if a new CIM_ResourceAllocationSettingData instance is created for elements whose capabilities are defined by the associated CIM_AllocationCapabilities instance.
- 4 (“Supported“) indicates that the component CIM_ResourceAllocationSettingData instance represents a set of supported property values or the increments within a supported range of values.

7.1.1.3 ValueRange

The CIM Schema description of this property applies.

The possible values for the ValueRange property are as follows:

- 0 (“Point”) indicates that the component CIM_ResourceAllocationSettingData instance provides a single set of values.
- 1 (“Minimums”) indicates that this CIM_ResourceAllocationSettingData instance provides minimum values for numeric properties with a linear range. Unless restricted by a “Maximums” value on the same set of properties, all values that collate higher than the specified values are also considered to be supported by the associated capabilities instance.
- 2 (“Maximums”) indicates that this CIM_ResourceAllocationSettingData instance provides maximum values for numeric properties with a linear range. Unless restricted by a “Minimums” value on the same set of properties, all values that collate lower than the specified values are also considered to be supported by the associated capabilities instance.
- 3 (“Increments”) indicates that this CIM_ResourceAllocationSettingData instance provides increment values for numeric properties. These values represent the respective increment between the maximum and minimum values of the supported numeric settings.

7.2 Implementation

This subclause details the requirements related to the arrangement of instances and properties of those instances for implementations of the profile described in this clause.

Each instance of CIM_AllocationCapabilities shall be associated with one or more instances of CIM_ManagedElement through the CIM_ElementCapabilities association class.

Each instance of CIM_AllocationCapabilities shall be associated with zero or more instances of CIM_ResourceAllocationSettingData through the CIM_SettingsDefineCapabilities association class. The ResourceType property for each instance of CIM_ResourceAllocationSettingData associated with an instance of CIM_AllocationCapabilities through the CIM_SettingsDefineCapabilities association shall have the same value as the ResourceType property of the CIM_AllocationCapabilities instance.

If a CIM_ResourceAllocationSettingData instance has an associated CIM_AllocationCapabilities instance to represent the mutability of its properties and no non-null values are found for a property in the instance or instances of CIM_ResourceAllocationSettingData associated to the CIM_AllocationCapabilities instance with the CIM_SettingsDefineCapabilities association that property is not mutable.

If multiple CIM_AllocationCapabilities instances are associated through the CIM_ElementCapabilities association class to a single instance of CIM_ManagedElement, each instance of CIM_AllocationCapabilities and associated CIM_ResourceAllocationSettingData instances shall define one correlated set. A setting from one set shall not be combined with a setting from another set to define a valid allocation request for a given resource.

7.2.1 Default class CIM_AllocationCapabilities (optional)

The default CIM_AllocationCapabilities instance of a given resource type associated with a managed element may be modeled. This subclause describes the behavioral requirements if a default CIM_AllocationCapabilities instance is modeled.

The CIM_ElementCapabilities instance that associates the CIM_AllocationCapabilities instance of a given resource type to a managed element that represents a default CIM_AllocationCapabilities instance shall be implemented as specified in 7.5.3. Each instance of CIM_ManagedElement shall be referenced by at most one instance of CIM_ElementCapabilities as specified in 7.5.3 for a given resource type. This implies that at most a single default CIM_AllocationCapabilities instance for a given resource type may be associated to a manage element.

7.2.2 Modeling default settings (optional)

The default resource allocation settings for a CIM_AllocationCapabilities instance associated with a managed element may be modeled. This subclause describes the behavioral requirements if the default allocation settings are modeled.

The CIM_SettingsDefineCapabilities instance that associates the CIM_ResourceAllocationSettingData representing the default settings with the CIM_AllocationCapabilities instance shall be implemented as specified in 7.5.5. Each instance of CIM_AllocationCapabilities shall be referenced by at most one instance of CIM_SettingsDefineCapabilities implemented as specified in 7.5.5. This implies that at most a single default resource allocation settings exists and is modeled with a single instance of CIM_ResourceAllocationSettingData.

7.2.3 Modeling minimum settings (optional)

The minimum resource allocation settings for a CIM_AllocationCapabilities instance associated with a managed element may be modeled. This subclause describes the behavioral requirements if the minimum allocation settings are modeled.

The CIM_SettingsDefineCapabilities instance that associates the CIM_ResourceAllocationSettingData representing the minimum settings with the CIM_AllocationCapabilities instance shall be implemented as specified in 7.5.6. Each instance of CIM_AllocationCapabilities shall be referenced by at most one instance of CIM_SettingsDefineCapabilities implemented as specified in 7.5.6. This implies that at most a single minimum resource allocation settings exists and is modeled with a single instance of CIM_ResourceAllocationSettingData.

7.2.4 Modeling maximum settings (optional)

The maximum resource allocation settings for a CIM_AllocationCapabilities instance associated with a managed element may be modeled. This subclause describes the behavioral requirements if the maximum allocation settings are modeled.

The CIM_SettingsDefineCapabilities instance that associates the CIM_ResourceAllocationSettingData representing the maximum settings with the CIM_AllocationCapabilities instance shall be implemented as specified in 7.5.7. Each instance of CIM_AllocationCapabilities shall be referenced by at most one instance of CIM_SettingsDefineCapabilities implemented as specified in 7.5.7. This implies that at most a single maximum resource allocation settings exists and is modeled with a single instance of CIM_ResourceAllocationSettingData.

7.2.5 Modeling increment settings (optional)

The increment resource allocation settings for a CIM_AllocationCapabilities instance associated with a managed element may be modeled. This subclause describes the behavioral requirements if the increment allocation settings are modeled.

The CIM_SettingsDefineCapabilities instance that associates the CIM_ResourceAllocationSettingData representing the increment settings with the CIM_AllocationCapabilities instance shall be implemented as specified in 7.5.8. Each instance of CIM_AllocationCapabilities shall be referenced by at most one instance of CIM_SettingsDefineCapabilities implemented as specified in 7.5.8.This implies that at most a single increment resource allocation settings exists and is modeled with a single instance of CIM_ResourceAllocationSettingData.

7.2.6 Modeling supported point settings (optional)

The supported point resource allocation settings for a CIM_AllocationCapabilities instance associated with a managed element may be modeled. This subclause describes the behavioral requirements if the supported allocation settings for a managed element are modeled.

The CIM_SettingsDefineCapabilities instance that associated the CIM_ResourceAllocationSettingData representing the supported point settings with the CIM_AllocationCapabilities instance shall be implemented as specified in 7.5.9.

7.3 Methods

This subclause details the requirements for supporting intrinsic operations and extrinsic methods for the CIM elements defined by the profile described in this clause.

7.3.1 Profile conventions for operations

For each profile class (including associations), the implementation requirements for operations, including for those in the following default list, are specified in class-specific subclauses of this clause.

The default list of operations for all classes is as follows:

- GetInstance( )
- EnumerateInstances( )
- EnumerateInstanceNames( )

For classes that are referenced by an association, the default list also includes

- Associators( )
- AssociatorNames( )
- References( )
- ReferenceNames( )

7.3.2 CIM_AllocationCapabilities

All operations in the default list in 7.3.1 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

7.3.3 CIM_ResourceAllocationSettingData

All operations in the default list in 7.3.1 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

7.3.4 CIM_SettingsDefineCapabilities

Table 90 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 90, all operations in the default list in 7.3.1 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

Table 90 – Operations: CIM_SettingsDefineCapabilities

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

7.3.5 CIM_ElementCapabilities

Table 91 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in

Table 91, all operations in the default list in 7.3.1 shall be implemented as defined in DSP0200.

NOTE Related profiles may define additional requirements on operations for the profile class.

Table 91 – Operations: CIM_ElementCapabilities

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

7.4 Use cases

This subclause contains object diagrams and use cases for the DMTF Allocation Capabilities Profile. Use cases are informative and are not intended to define the requirements for conformance.

CIM_AllocationCapabilities and its associated set of CIM_ResourceAllocationSettingData instances define the allocation capability for a given resource type of a host system or resource pool, or describes the mutability and the valid ranges for change of a CIM_ResourceAllocationSettingData instance. Figure 14 demonstrates uses of CIM_AllocationCapabilities to represent the allocation capabilities of a virtualization system and one resource pool within the virtualization system. Figure 15 demonstrates the use of CIM_AllocationCapabilities to represent the mutability of a property within an associated Current CIM_ResourceAllocationSettingData.

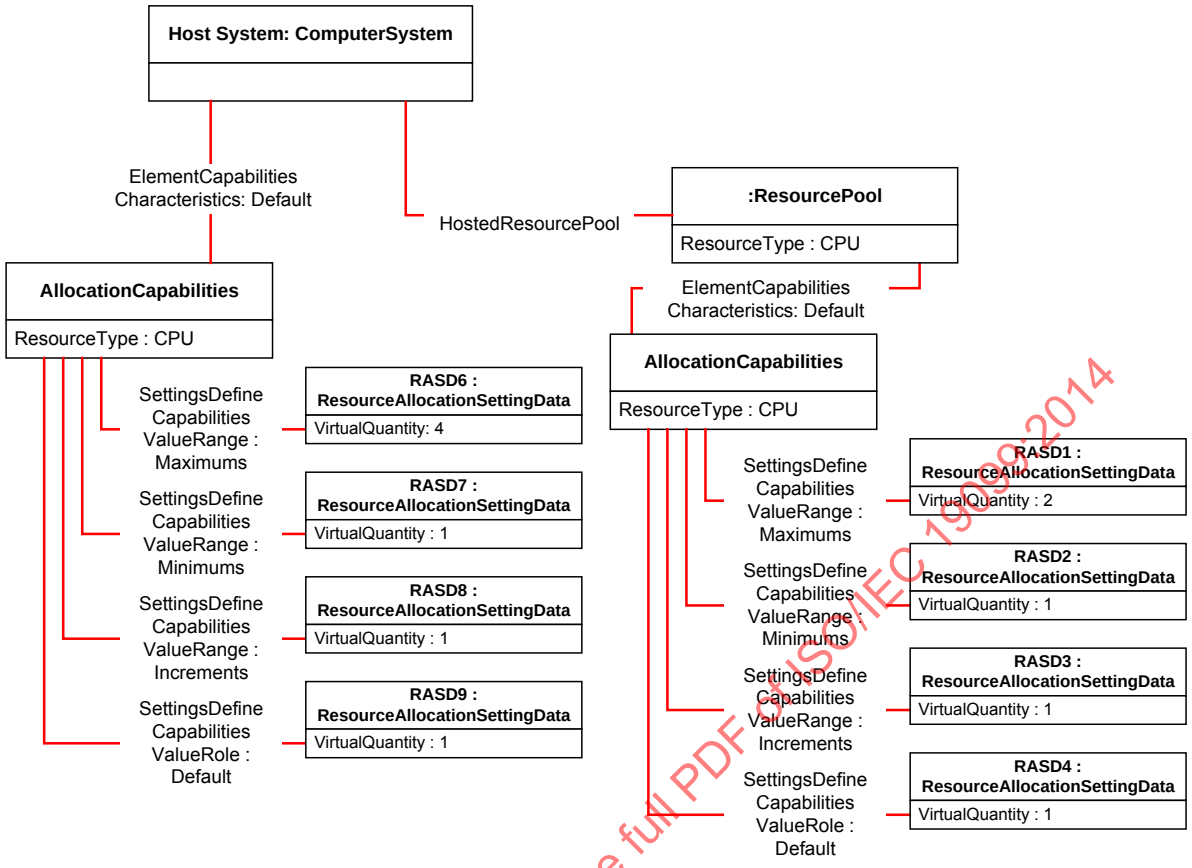

Figure 14 – Allocation capabilities associated to CIM_ComputerSystem and CIM_ResourcePool

7.4.1 Associating CIM_AllocationCapabilities with a host system

The CIM_AllocationCapabilities instance associated to the host system defines the allocation capabilities of the host system for a resource type. In Figure 14 the host system is capable of accepting allocation requests for systems with up to 4 CPUs and a minimum of one CPU or if no value for CPU VirtualQuantity is specified in an allocation request the default value of 1 is used.

7.4.2 Associating CIM_AllocationCapabilities with a resource pool

The CIM_AllocationCapabilities instance associated to the resource pool defines the allocation capabilities of the resource pool. This usage allows a host system to present the capabilities of multiple resource pools for a resource type. As illustrated in Figure 14, the capability set for this specific CPU resource pool subsets the overall capabilities of the virtualization system. The specific resource pool instance in the figure limits the maximum CPUs in a virtual system to two.

7.4.3 Associating CIM_AllocationCapabilities with a CIM_ResourceAllocationSettingData instance

As shown in Figure 15, the instance of CIM_AllocationCapabilities that is associated to the current CIM_ResourceAllocationSettingData instances specifies the mutability of the VirtualQuantity property in the associated Current CIM_ResourceAllocationSettingData instance. Figure 15 shows a capability set of a current CIM_ResourceAllocationSettingData associated to a single instance of CIM_Processor. This specifies that the VirtualQuantity property in the Current CIM_ResourceAllocationSettingData can be changed from 1 to 2.

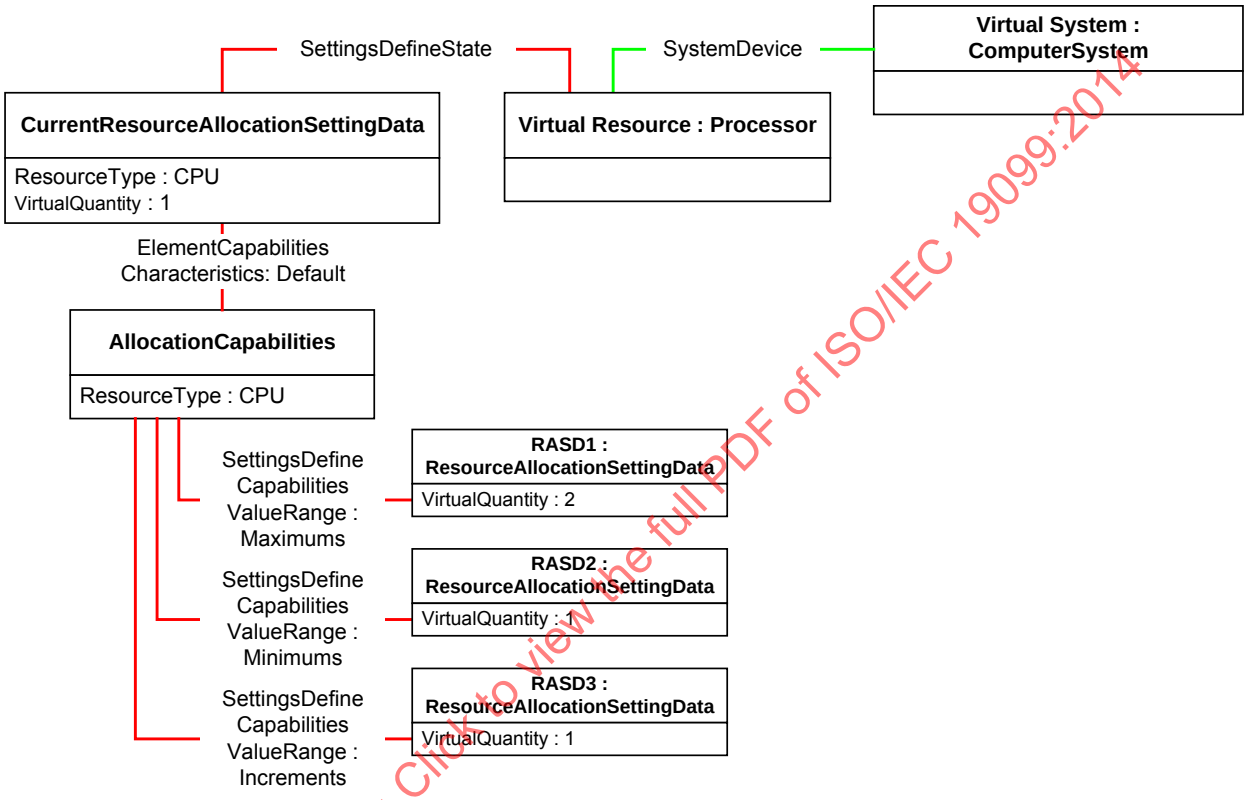

Figure 15 – Allocation capabilities associated to CIM_ResourceAllocationSettingData

7.4.4 Associating multiple CIM_AllocationCapabilities with one resource pool instance

Figure 16 shows an example in which multiple capability sets are used to express multiple capabilities of a single resource pool. This example shows a system that allows either shared or exclusive access of a resource. Based on the type of allocation selected the Default, Minimum and Increment CIM_ResourceAllocationSettingData instances reflect different property values. Each capability set defines one correlated set. A setting from one set may not be combined with a setting from another set to form a valid allocation request for a given resource.

While this example is based on the SharingMode property in CIM_AllocationCapabilities, other possibilities exist. For example, different capability sets may be based on the AllocationUnits property in CIM_ResourceAllocationSettingData.

The property CIM_ElementCapabilities.Characteristics set to 2 (“Default”) shows that the CIM_AllocationCapabilities instance AC0 represents the default capability set for this resource pool.

This same pattern applies to capability sets associated with a host system or CIM_ResourceAllocationSettingData, as well as to the example shown in Figure 16.

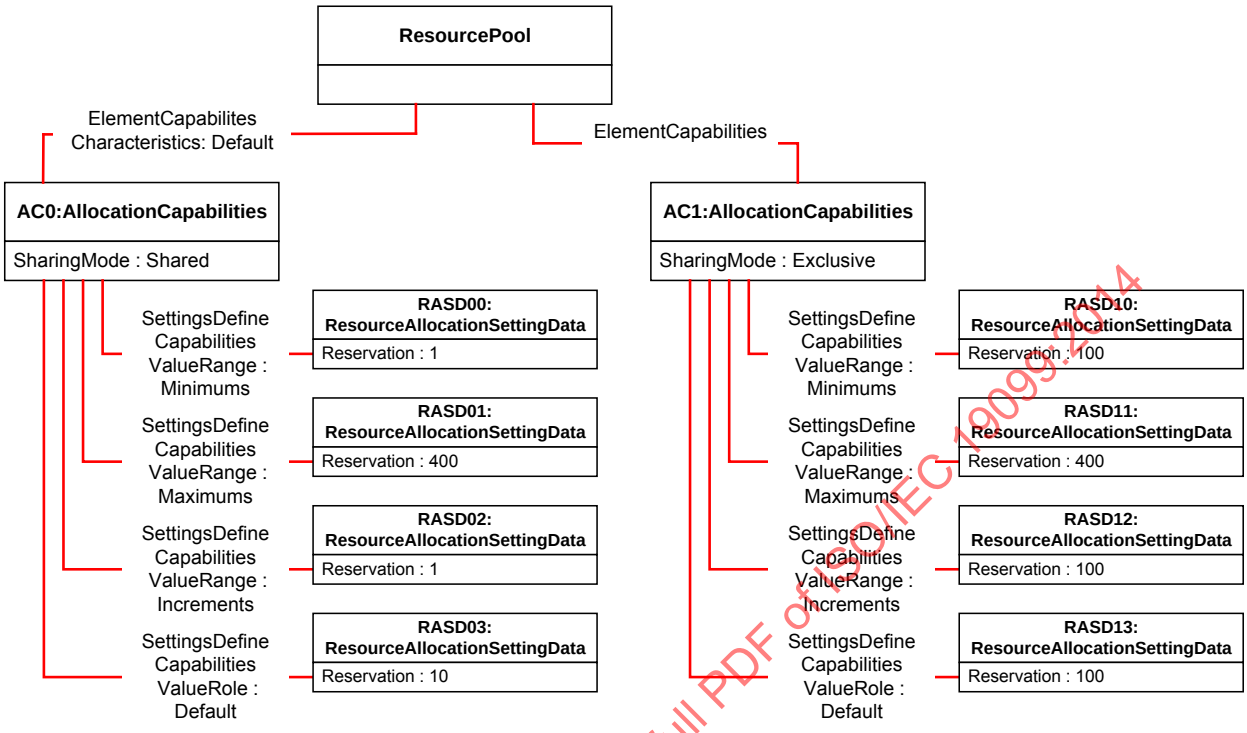

Figure 16 – Multiple CIM_AllocationCapabilities instances

7.4.5 Discovering a host system's allocation capability for a given resource type

The client can enumerate the CIM_AllocationCapabilities instances associated to the target CIM_ComputerSystem (host system) with the CIM_ElementCapabilities association, filtering on the CIM_AllocationCapabilities.ResourceType property to select only the CIM_AllocationCapabilities instances of the desired resource type.

7.4.6 Discovering the allocation capability for a given resource type for a specific resource pool

The client enumerates the CIM_ResourcePool instances by filtering on the CIM_ResourcePool.ResourceType property to select only the CIM_ResourcePool instances of the desired resource type. For each of the target instances of CIM_ResourcePool select all of the instances of CIM_AllocationCapabilities associated with the CIM_ElementCapabilities association.

7.4.7 Determining the default instance of CIM_AllocationCapabilities for a given resource type

From the selected CIM_AllocationCapabilities instance(s) (see 7.4.5 and 7.4.6) the client selects the CIM_AllocationCapabilities instance associated through the CIM_ElementCapabilities association where the value of the CIM_ElementCapabilities.Characteristics property is 2 ("Default").

7.4.8 Determining the default, supported point, and valid ranges of property values representing the allocation capability from a selected instance of CIM_AllocationCapabilities

The client finds the CIM_ResourceAllocationSettingData instance associated with a selected instance of CIM_AllocationCapabilities (see 7.4.5 and 7.4.6) through the CIM_SettingsDefineCapabilities association where the value of the CIM_SettingsDefineCapabilities.ValueRole property is 0 (“Default”). The values within the selected CIM_ResourceAllocationSettingData instance represent the default values for the host system or the selected resource pool. A null value specifies that the property is not relevant for the resource type.

After determining that there is a non-null default property value, the client finds the CIM_ResourceAllocationSettingData instances associated with a selected instance of CIM_AllocationCapabilities (see 7.4.5 and 7.4.6) through the CIM_SettingsDefineCapabilities association where the value of the CIM_SettingsDefineCapabilities.ValueRole property is 3 (“Supported”) and the value of the CIM_SettingsDefineCapabilities.ValueRange property is 0 (“Point”). The set of non-null values for a given property within the selected set of CIM_ResourceAllocationSettingData represents supported values for that property.

For numeric properties the client finds the CIM_ResourceAllocationSettingData instances associated with a selected instance of CIM_AllocationCapabilities (see 7.4.5 and 7.4.6) through the CIM_SettingsDefineCapabilities associations where the value of the CIM_SettingsDefineCapabilities.ValueRole property is 1 (“Minimums”), 2 (“Maximums”), or 3 (“Increments”). The minimum and maximum values define the range of valid parameters. The increment values describe the valid steps within a specified range. Each of these instances represents a limitation on the range of supported values. For example, if a property has a minimum value but no maximum value specified, the maximum is not limited. If the property has a maximum value but no minimum value there is no minimum value.

7.4.9 Discovering the supported changes of a property value in an instance of a CIM_ResourceAllocationSettingData

The client enumerates the set of CIM_AllocationCapabilities instances associated with the association CIM_ElementCapabilities to the target CIM_ResourceAllocationSettingData instance.

The client finds the CIM_ResourceAllocationSettingData instance associated with a selected instance of CIM_AllocationCapabilities through the CIM_SettingsDefineCapabilities association where the value of the CIM_SettingsDefineCapabilities.ValueRole property is 3 (“Supported”) and the value of the ValueRange property is 0 (“Point”). The set of non-null values for a given property within the selected set of CIM_ResourceAllocationSettingData represents supported point values for that property.

For numeric properties the client finds the CIM_ResourceAllocationSettingData instances associated with a selected instance of CIM_AllocationCapabilities through the CIM_SettingsDefineCapabilities associations where the value of the CIM_SettingsDefineCapabilities.ValueRole property is 1 (“Minimums”), 2 (“Maximums”), or 3 (“Increments”). The minimum and maximum values define a range of supported values for numeric properties. The increments value describes the valid steps within a specified range. Each of these instances represents a limitation on the range of supported values. For example, if a property has a minimum value but no maximum value specified, the maximum is not limited. If the property has a maximum value but no minimum value specified, the minimum is not constrained.

7.5 CIM elements

Table 92 shows the instances of CIM Elements for the profile described in this clause. Instances of the CIM Elements shall be implemented as described in Table 92. Subclauses 7.2 (“Implementation”) and 7.3 (“Methods”) may impose additional requirements on these elements.

Table 92 – CIM elements: Allocation Capabilities Profile

Element name	Requirement	Description
Classes		
CIM_AllocationCapabilities	Mandatory	See 7.5.1.
CIM_ElementCapabilities	Mandatory	See 7.5.2.
CIM_SettingsDefineCapabilities	Mandatory	See 7.5.3, 7.5.5, 7.5.6, 7.5.7, 7.5.8, and 7.5.9.
Indications		
None defined in the profile described in this clause		

7.5.1 CIM_AllocationCapabilities

CIM_AllocationCapabilities represents the allocation capabilities of a host system or resource pool or represents the mutability a CIM_ResourceAllocationSettingData instance.

Table 93 provides information about the properties of CIM_AllocationCapabilities.

Table 93 – Class: CIM_AllocationCapabilities

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
ResourceType	Mandatory	None
OtherResourceType	Conditional	Shall be used if ResourceType matches 1 (“Other”).
RequestTypesSupported	Mandatory	None
SharingMode	Mandatory	None
SupportedAddStates	Optional	None
SupportedRemoveStates	Optional	None

7.5.2 CIM_ElementCapabilities

CIM_ElementCapabilities associates an instance of CIM_AllocationCapabilities with a subclass of CIM_ManagedElement assumed to be CIM_System, CIM_ResourcePool, or CIM_ResourceAllocationSettingData.

Table 94 defines the properties of CIM_ElementCapabilities.

Table 94 – Class: CIM_ElementCapabilities

Properties	Requirement	Notes
ManagedElement	Mandatory	**Key** Cardinality 1..*
Capabilities	Mandatory	**Key** Shall be a reference to the CIM_AllocationCapabilities instance Cardinality *
Characteristics	Mandatory	

7.5.3 CIM_ElementCapabilities (default)

CIM_ElementCapabilities associates an instance of CIM_AllocationCapabilities with a subclass of CIM_ManagedElement assumed to be CIM_System, CIM_ResourcePool, or CIM_ResourceAllocationSettingData representing the default capabilities.

Table 95 defines the properties of CIM_ElementCapabilities.

Table 95 – Class: CIM_ElementCapabilities (default)

Properties	Requirement	Notes
ManagedElement	Mandatory	**Key** Cardinality 1
Capabilities	Mandatory	**Key** Shall be a reference to a default CIM_AllocationCapabilities instance Cardinality 1
Characteristics	Mandatory	**Matches 2 “Default”**

7.5.4 CIM_SettingsDefineCapabilities

CIM_SettingsDefineCapabilities associates a CIM_ResourceAllocationSettingData instance, representing the settable or mutable allocation settings for a resource, with a CIM_AllocationCapabilities instance.

Table 96 provides information about the properties of CIM_SettingsDefineCapabilities.

Table 96 – Class: CIM_SettingsDefineCapabilities

Elements	Requirement	Notes
GroupComponent	Mandatory	Shall be a reference to an instance of CIM_AllocationCapabilities Cardinality 1
PartComponent	Mandatory	Shall be a reference to an instance of CIM_ResourceAllocationSettingData Cardinality 1..*
PropertyPolicy	Mandatory	
ValueRole	Mandatory	
ValueRange	Mandatory	

7.5.5 CIM_SettingsDefineCapabilities – Default

CIM_SettingsDefineCapabilities associates a CIM_ResourceAllocationSettingData instance, representing default allocation settings for a resource, with a CIM_AllocationCapabilities instance.

Table 97 provides information about the properties of CIM_SettingsDefineCapabilities (Default).

Table 97 – Class: CIM_SettingsDefineCapabilities (Default)

Elements	Requirement	Notes
GroupComponent	Mandatory	Shall be a reference to an instance of CIM_AllocationCapabilities Cardinality 1
PartComponent	Mandatory	Shall be a reference to an instance of CIM_ResourceAllocationSettingData Cardinality 1
PropertyPolicy	Mandatory	Matches 0 (“Independent”)
ValueRole	Mandatory	Matches 0 (“Default”)
ValueRange	Mandatory	Matches 0 (“Point”)

7.5.6 CIM_SettingsDefineCapabilities (minimums)

CIM_SettingsDefineCapabilities associates a CIM_ResourceAllocationSettingData instance representing the minimum values of valid numeric settings to a CIM_AllocationCapabilities instance.

Table 98 provides information about the properties of CIM_SettingsDefineCapabilities (Minimums).

Table 98 – Class: CIM_SettingsDefineCapabilities (minimums)

Elements	Requirement	Notes
GroupComponent	Mandatory	Shall be a reference to an instance of CIM_AllocationCapabilities Cardinality 1
PartComponent	Mandatory	Shall be a reference to an instance of CIM_ResourceAllocationSettingData Cardinality 1
PropertyPolicy	Mandatory	Matches 0 (“Independent”)
ValueRole	Mandatory	Matches 3 (“Supported”)
ValueRange	Mandatory	Matches 1 (“Minimums”)

7.5.7 CIM_SettingsDefineCapabilities (maximums)

CIM_SettingsDefineCapabilities associates a CIM_ResourceAllocationSettingData instance representing the maximum values of valid numeric settings to a CIM_AllocationCapabilities instance.

Table 99 provides information about the properties of CIM_SettingsDefineCapabilities (Maximums).

Table 99 – Class: CIM_SettingsDefineCapabilities (maximums)

Elements	Requirement	Notes
GroupComponent	Mandatory	Shall be a reference to an instance of CIM_AllocationCapabilities Cardinality 1
PartComponent	Mandatory	Shall be a reference to an instance of CIM_ResourceAllocationSettingData Cardinality 1
PropertyPolicy	Mandatory	Matches 0 (“Independent”)
ValueRole	Mandatory	Matches 3 (“Supported”)
ValueRange	Mandatory	Matches 2 (“Maximums”)

7.5.8 CIM_SettingsDefineCapabilities – Increments

CIM_SettingsDefineCapabilities associates a CIM_ResourceAllocationSettingData instance, representing the increment between the maximum and minimum values of supported numeric settings, to a CIM_AllocationCapabilities instance.

Table 100 provides information about the properties of CIM_SettingsDefineCapabilities (Increments).

Table 100 – Class: CIM_SettingsDefineCapabilities (Increments)

Elements	Requirement	Notes
GroupComponent	Mandatory	Shall be a reference to an instance of CIM_AllocationCapabilities Cardinality 1
PartComponent	Mandatory	Shall be a reference to an instance of CIM_ResourceAllocationSettingData Cardinality 1
PropertyPolicy	Mandatory	Matches 0 (“Independent”)
ValueRole	Mandatory	Matches 3 (“Supported”)
ValueRange	Mandatory	Matches 3 (“Increments”)

7.5.9 CIM_SettingsDefineCapabilities – Supported Point

CIM_SettingsDefineCapabilities associates a CIM_ResourceAllocationSettingData instance, representing the settable or mutable allocation settings for a resource, with a CIM_AllocationCapabilities instance.

Table 101 provides information about the properties of CIM_SettingsDefineCapabilities (Independent Supported Point).

Table 101 – Class: CIM_SettingsDefineCapabilities (Independent Supported Point)

Elements	Requirement	Notes
GroupComponent	Mandatory	Shall be a reference to an instance of CIM_AllocationCapabilities Cardinality 1
PartComponent	Mandatory	Shall be a reference to an instance of CIM_ResourceAllocationSettingData Cardinality 1..*
PropertyPolicy	Mandatory	Matches 0 (“Independent”) or 1 (“Correlated”)
ValueRole	Mandatory	Matches 3 (“Supported”)
ValueRange	Mandatory	Matches 0 (“Point”)

8 Processor Resource Virtualization Profile

Profile Name: Processor Resource Virtualization

Version: 1.0.0

Organization: DMTF

CIM schema version: 2.21

Central Class: CIM_ResourcePool

Scoping Class: CIM_System

The Processor Resource Virtualization Profile is a component profile that defines the minimum object model needed to provide for the CIM representation and management of the virtualization of processors.

Table 102 lists other profiles that the Processor Resource Virtualization Profile depends on, or that may be used in context of this profile.

Table 102 – Related profiles for the Processor Resource Virtualization Profile

Profile name	Organization	Version	Relationship	Description
Resource Allocation	DMTF	1.1	Specializes	The profile that adds the capability to represent the allocation of resources to consumers. See clause 5.
Allocation Capabilities	DMTF	1.0	Specializes	The profile that describes the default property values, supported property values, and range of property values for a resource allocation request. See clause 7.
Profile Registration	DMTF	1.0	Mandatory	The profile that specifies registered profiles. See DSP1033.
CPU	DMTF	1.0	Optional	The profile that adds the capability to represent processors in a managed system. See 8.2.2 and DSP1022.

The *CPU Profile* lists additional related DMTF management profiles; these relationships are not further specified in the profile described in this clause.

8.1 Description

This subclause introduces the management domain addressed by the profile described in this clause, and outlines the central modeling elements established for representation and control of the management domain.

8.1.1 General

In computer virtualization systems, virtual computer systems are composed of component virtual resources. The profile described in this clause specifies the allocation and management of host processor resources in support of virtual processors. From an operating system or application viewpoint virtual processors are functionally equivalent to "physical" processors.

The profile described in this clause applies the resource virtualization pattern defined in the Resource Allocation Profile (see clause 5) and the allocation capabilities pattern defined in the Allocation

Capabilities Profile (see clause 7) to enable the management of processor resources that are allocated to virtual systems. The profile described in this clause defines additional CIM elements and constraints beyond those defined in the specialized profiles. Optionally, implementations may implement DSP1022 to represent host processors.

8.1.2 Processor resource virtualization class schema

Figure 17 represents the class schema of the profile described in this clause. It outlines the elements that are adapted by the profile described in this clause. For simplicity, the prefix CIM_ has been removed from the name of the classes.

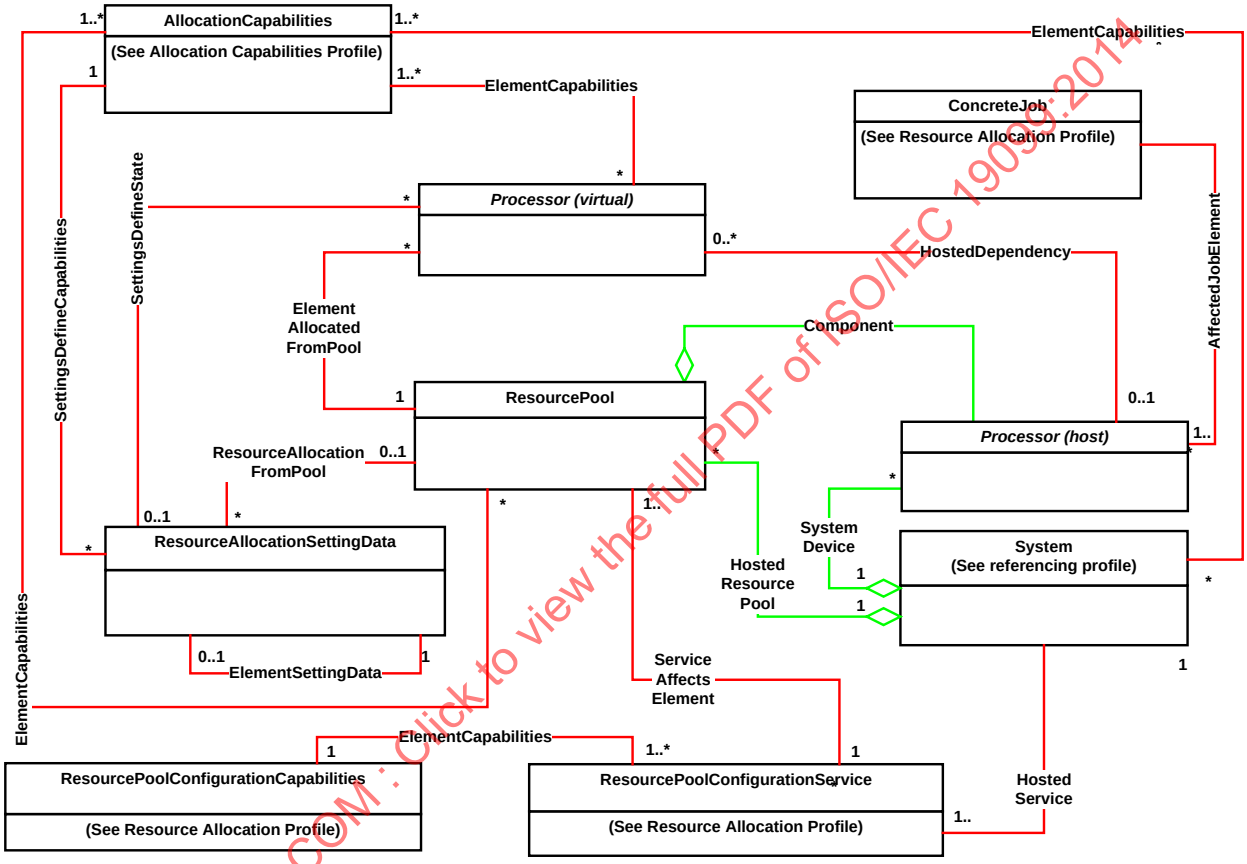

Figure 17 – Processor Resource Virtualization Profile: Class Diagram

The profile described in this clause specifies the use of the following classes and associations:

- The CIM_ResourcePool class models resource pools for processor resources. Host processor resources are allocated from their resource pool and used to create the virtual processors for virtual systems.
- The CIM_Component association models the relationship between processor resource pools and host processors as components of the resource pools.
- The CIM_ElementAllocatedFromPool association models hierarchies of processor resource pools and modeling the relationship of processor resource pools and the virtual processors allocated from those.

- The CIM_HostedResourcePool association models the hosting dependency between a processor resource pool and its host system. A host system supports at least one processor resource pool.
- The CIM_Processor class models the following kinds of processors:
  - processors as a device in the scope of a system, as modeled by the CIM_SystemDevice association
  - processors as a result of a processor resource allocation from a resource pool, as modeled by the CIM_ElementAllocatedFromPool association
  - processors as a component within processor resource pools, as modeled through the CIM_Component association
- The CIM_ResourceAllocationSettingData class models processor resource allocations or processor resource allocation requests.
- The CIM_AllocationCapabilities class and the CIM_ElementCapabilities association model
  - the processor resource allocation capabilities of host systems
  - the processor resource allocation capabilities of processor resource pools
  - the mutability of existing processor resource allocations
- The CIM_SettingsDefineCapabilities association models the relation between processor resource allocation capabilities and the settings that define these capabilities.
- The CIM_ResourcePoolConfigurationService models configuration services for processor resource pools and the CIM_ResourcePoolConfigurationCapabilities class modeling their capabilities.
- The CIM_ConcreteJob class and the CIM_AffectedJobElement association models asynchronous management tasks initiated through memory resource pool configuration services.

In general, any mention of a class in this document means the class itself or its subclasses. For example, a statement such as "an instance of the CIM_Processor class" implies an instance of the CIM_Processor class or a subclass of the CIM_Processor class.

8.1.3 Resource pools

The profile described in this clause applies the concept of resource pools defined in the Resource Allocation Profile (see clause 5) to the resource type 3 (Processor).

8.1.3.1 General

The profile described in this clause uses processor resource pools as the focal point for processor allocations. A processor resource pool represents an aggregate amount of processing capacity, and keeps track of the amount of processor capacity that has been allocated to consumers such as virtual systems. A resource pool also defines the scope in which relative weights are interpreted. A resource pool represents a part or all of the aggregated processing power of a virtualization platform in an abstract sense, that is, it represents the sum of the processing power of the host resources aggregated into the resource pool and is expressed in allocation units such as MHz, percentage, or count of processors.

Note that the resource type of a resource pool governs the type of the resources that are allocated from the resource pool. Opposed to that the resource type of the resources that are aggregated by the resource pool may differ from the resource type of the pool. For example, a resource pool with a resource type of 3 (Processor) supports the allocation of virtual processors. However, the resources that are aggregated by that resource pool may be of a different type; for example, that resource pool might

aggregate processor fragments, or it might simply represent a certain amount of processing power without representing individual processors.

8.1.3.2 Representation of host resources

A processor resource pool represents an aggregated amount of processing power provided by host resources that enables the allocation of virtual processors. However the explicit representation of the host resources aggregated by a resource pool is optional: In some cases implementations may explicitly represent the host resources such as for example host processors. In other cases implementations may choose not to explicitly represent the host resources aggregated by a resource pool, that is, an implementation may choose to model a resource pool as the sole model element that represents host processing capacity for the support of (allocated) virtual processors, but not detail the host resources that provide that processing capacity.

The processor resources of a host system can be represented by a single processor resource pool, with capacity equal to the processing capacity of the host computer system. The processor resources of a host computer system may be represented by multiple resource pools, providing more flexible control over the allocation of processor resources to virtual systems.

8.1.3.3 Hierarchies of processor resource pools

The profile described in this clause applies the concept of resource pool hierarchies defined in the Resource Allocation Profile (see clause 5) to the processor resource type; see 5.1.1.4.

8.1.4 Resource allocation

The profile described in this clause applies the concept of device resource allocation defined in the Resource Allocation Profile (see clause 5) to the processor resource type; see 5.1.3.

8.1.4.1 Processor resource allocation request

The processor requirements of a virtual system are defined as part of the "defined" virtual system configuration; see the Virtual System Profile in clause 12 for a definition of the "defined" virtual system configuration. The "defined" virtual system configuration contains processor resource allocation requests represented as RASD instances.

8.1.4.2 Processor resource allocation

As a virtual system is activated (or instantiated), one or more virtual processors need to be allocated as requested by processor resource allocation requests in the virtual system definition. Processor resource allocations are represented as RASD instances in the "state" virtual system configuration. The number of discrete processors exposed to the virtual system is specified by the value of the VirtualQuantity property.

The central properties describing a processor resource allocation request or a processor resource allocation are Reservation, Limit, and Weight.

The values of both the Reservation and Limit properties are specified in allocation units as expressed by the value of the AllocationUnits property. Possible allocation units are a frequency (for example, "Hertz" or "MHz"), a percentage (with respect to some base value such as the sum of all available processing power), or a count (expressing a number of processors or processor fractions to be allocated).

The value of the Reservation property specifies a lower bound on the quantity of host processor resources available for the virtual computer system that are guaranteed to be available for use. If a virtual computer system does not consume its full allocation of reserved resources, the host system may allow its unused portion to be utilized by other virtual computer systems. The value of the Limit property specifies a limit or upper bound on the quantity of host processor resources that may be consumed by the

virtual computer system. A limit is an artificial cap that may not be exceeded, even if otherwise-idle host processor resources are available.

The value of the Weight property specifies the relative importance of the set of allocated virtual processors, and is expressed in abstract numeric units. A virtual system is entitled to consume host processor resources at a rate that is directly proportional to the value of the Weight property.

8.1.4.3 Virtual processors

A virtual processor is the instantiation of allocated processor resources that is exposed to a virtual system through a logical processor device; it is the result of the processor resource allocation based on a processor resource allocation request. A virtual processor may be realized using techniques such as time sharing, but may also be a host processor that is directly passed through to the virtual system.

A virtual processor is represented by a CIM_Processor instance that is part of the virtual system representation.

8.1.4.4 Dedicated processors

A dedicated host processor is a processor owned by the host system that is exclusively reserved for support of a virtual processor of a particular virtual system.

8.1.4.5 Consumption of host processing power

A processor resource allocation request references the processor resource pool to be used by specifying the value of the PoolID property. Host processor resources are allocated from the identified processor resource pool during processor resource allocation.

8.1.4.5.1 Statically controlled processor resource consumption

With statically controlled processor resource consumption, a particular processor resource allocation request is granted only if the amount of host processing resource available in the addressed processor resource pool is at least as large as the amount requested. This approach is called admission control. Each successfully allocated processor resource reduces the amount of processing resources available from the pool, respectively. In the CIM representation of the processor resource pool, the amount of processing power assigned to consumers from the pool is visible through the value of the Reserved property. With admission control the processing capacity represented by a particular resource pool remains larger than the sum of all reservations out of that pool. Admission control checks are typically performed for the following operations: creating a resource pool, powering on a virtual system, and modifying the resource allocation settings for either a resource pool or a running virtual system.

8.1.4.5.2 Dynamically controlled processor resource consumption

With dynamically controlled processor resource consumption, the amount of host processing resources allocated in support of a virtual processor is not a constant value but varies significantly over time, depending on factors like the processing requirements of the software executed within the virtual system or the processing power consumption of other virtual machines.

Further, with dynamically controlled processor resource consumption, the amount of processing power managed through a processor resource pool conceptually may be considered as unlimited, such that no admission control is performed at the time virtual processors are initially allocated (usually at virtual system activation time). Of course, this approach may result in an over commitment situation where the host system experiences processing power shortages at a later point in time, such that ultimately the host system is no longer able to support the sum of processing power requests of all hosted virtual systems.

8.2 Implementation

This subclause provides normative requirements related to the arrangement of instances and properties of instances for implementations of the profile described in this clause.

8.2.1 Allocation units

This subclause details requirements for the unit of measurement that applies to the specification of processor resource allocations.

8.2.1.1 General

Processor resource allocations and processor resource allocation requests shall be expressed through DSP0004 programmatic units using one these base units: "hertz", "percent", or "count".

8.2.1.2 Use of the base unit "hertz"

If the base unit of "hertz" is used, the following provisions apply:

- CIM_Processor class
  - MaxClockSpeed property: is expressed in MHz as defined in the CIM Schema
  - CurrentClockSpeed property: is expressed in MHz as defined in the CIM Schema
- CIM_ResourceAllocationSettingData class
  - AllocationUnits property: Value shall use a base unit of "hertz".
  - Reservation property: Value shall be expressed in the unit expressed by the value of the AllocationUnits property.
  - Limit property: Value shall be expressed in the unit expressed by the value of the AllocationUnits property.
- CIM_ResourcePool class
  - AllocationUnits property: Value shall use a base unit of "hertz".
  - Capacity property: Value shall be expressed in the unit expressed by the value of the AllocationUnits property.
  - Reserved property: Value shall be expressed in the unit expressed by the value of the AllocationUnits property.

8.2.1.3 Use of the base unit "percent"

If the base unit of "percent" is used, the following provisions apply:

- CIM_Processor class
  - MaxClockSpeed: Value is expressed in MHz as defined in the CIM Schema
  - CurrentClockSpeed: Value is expressed in MHz as defined in the CIM Schema
- CIM_ResourceAllocationSettingData class
  - AllocationUnits property: Value shall be "percent"
  - Reservation property: Value shall be expressed in percent, stating a minimum percentage of the processing power as requested from the resource pool identified by the RASD instance.
  - Limit property: Value shall be expressed in percent, stating a maximum percentage of the processing power to be provided by the resource pool identified by the RASD instance.

- CIM_ResourcePool class
  - AllocationUnits property: Value shall be "percent"
  - Capacity property: Value shall be expressed in the unit expressed by the value of the AllocationUnits property.
  - Reserved property: Value shall be expressed in percent, stating the reserved percentage of processing power presently allocated from the pool.

8.2.1.4 Use of the base unit "count"

If the base unit of "count" is used the following provisions apply:

- CIM_Processor class
  - MaxClockSpeed: Value is expressed in MHz as defined in the CIM Schema.
  - CurrentClockSpeed: Value is expressed in MHz as defined in the CIM Schema.
- CIM_ResourceAllocationSettingData class
  - AllocationUnits property: Value shall use a base unit of "count" the counted items shall be processors. For example, a unit of a tenth of a processor can be expressed using the value “count*0.1”.
  - Reservation property: Value shall be expressed in the unit expressed by the value of the AllocationUnits property.
  - Limit property: Value shall be expressed in the unit expressed by the value of the AllocationUnits property.
- CIM_ResourcePool class
  - AllocationUnits property: Value shall be use a base unit of "count" the counted items shall be processors. For example, a unit of a tenth of a processor can be expressed using the value “count*0.1”.
  - Capacity property: Value shall be expressed in the unit expressed by the value of the AllocationUnits property.
  - Reserved property: Value shall be expressed in the unit expressed by the value of the AllocationUnits property.

8.2.2 Host resources

The implementation of the representation of host processor resources is optional.

If the representation of host processors is implemented, the provisions in this subclause apply.

Each host processor shall be represented by exactly one CIM_Processor instance that is associated with the CIM_System instance that represents the host system through an instance of the CIM_SystemDevice association.

Implementations may implement DSP1022 for host processors.

8.2.3 Resource pools

This subclause adapts the CIM_ResourcePool class for the representation of processor resource pools.

8.2.3.1 ResourceType property

The value of the ResourceType property shall be 3 (Processor).

8.2.3.2 ResourceSubType property

The implementation of the ResourceSubType property is optional.

If the ResourceSubType property is implemented, the provisions in this subclause apply.

The value of the ResourceSubType property shall designate a resource subtype. The format of the value shall be as follows: "<org-id>:<org-specific>". The <org-id> part shall identify the organization that defined the resource subtype value; the <org-specific> part shall uniquely identify a resource subtype within the set of subtype defined by the respective organization.

8.2.3.3 Primordial property

The value of the Primordial property shall be set to TRUE for any CIM_ResourcePool instance that represents a primordial processor resource pool. For other CIM_ResourcePool instances that represent processor resource pools, the value of the Primordial property shall be set to FALSE.

8.2.3.4 PoolID property

The value of the PoolID property shall be set such that it enables unique identification of the CIM_ResourcePool instance within the scoping host system.

8.2.3.5 Reserved property

The implementation of the Reserved property is optional.

If the Reserved property is implemented, its value shall reflect the amount of host processing resource that is actually reserved from the resource pool, in units as expressed by the value of the AllocationUnits property (see 8.2.3.7).

8.2.3.6 Capacity property

The implementation of the Capacity property is conditional.

Condition: The Capacity property shall be implemented if the representation of the aggregation of host resources is implemented (see 8.2.4).

If the Capacity property is implemented, its value shall reflect the maximum amount of processing power that can be allocated from the resource pool, in units as expressed by the value of the AllocationUnits property (see 8.2.3.7). If the CIM_ResourcePool instance represents a processor resource pool with unlimited capacity, the value of the Capacity property shall be set to the largest value supported by the uint64 datatype.

The special value NULL shall be used if the implementation does not have knowledge about the resource capacity represented by the pool. This may reflect a permanent or a temporary situation.

8.2.3.7 AllocationUnits property

The value of the AllocationUnits property shall be expressed through DSP0004 programmatic units using one these base units: "hertz", "percent", or "count".

NOTE The units defined by the value of the AllocationUnits property apply to the values of the Reserved and the Limit property; it does not apply to the value of the VirtualQuantity property.

8.2.3.8 Instance requirements

Each processor resource pool shall be represented by a CIM_ResourcePool instance; the provisions of 8.5.12 and 8.2.3 apply. The CIM_ResourcePool instance shall be associated with the CIM_System instance representing the host system through an instance of the CIM_HostedResourcePool association (see the Resource Allocation Profile described in clause 5).

8.2.4 Aggregation of host resources

The implementation of the representation of the aggregation of host processor(s) into a processor resource pool is optional.

If the representation of the aggregation of host processors is implemented, it may be supported for all or for only for some processor resource pools. If the aggregation of host processors is supported for a particular resource pool, any instance of the CIM_Processor class representing a host processor that contributes processing resources into that resource pool shall be associated to the CIM_ResourcePool instance representing the resource pool through an instance of the subclass of CIM_Component association; the provisions of 8.5.1 apply.

8.2.5 Processor resource pool hierarchies

The implementation of the representation of processor resource pool hierarchies is optional. If implemented, any concrete processor resource pool shall be represented through a CIM_ResourcePool instance, where all of the following conditions shall be met:

- The value of the Primordial property shall be FALSE.
- The instance shall be associated through an instance of CIM_ElementAllocatedFromPool association to the CIM_ResourcePool instance that represents its parent processor resource pool.
- The instance shall be associated through an instance of the CIM_ElementSettingData association to the RASD instance that represents the amount of processing power allocated from the parent resource pool.

8.2.6 Default processor resource pool

The implementation of designating a default processor resource pool is optional. If implemented, all of the following conditions apply:

- The default processor resource pool shall be represented by a CIM_ResourcePool instance; see 8.2.8.3.2.
- That instance shall be associated to the CIM_AllocationCapabilities instance that represents the pools default allocation capabilities as specified in 8.2.8.4.3.
- The same CIM_AllocationCapabilities instance shall also represent the systems default allocation capabilities as specified in 8.2.8.3.2.

8.2.7 Processor resource pool management

The implementation of processor resource pool management is optional.

If implemented, the information in 5.2.4 applies; the profile described in this clause does not specify specializations or extensions of resource pool management beyond those defined by the Resource Allocation Profile (see clause 5).

8.2.8 Processor resource allocation

This subclause details requirements for the representation of resource allocation information through CIM_ResourceAllocationSettingData (RASD) instances.

8.2.8.1 General

NOTE The Resource Allocation Profile (see clause 5) specifies two alternatives for modeling resource allocation: *simple resource allocation* and *virtual resource allocation*.

Implementations of the profile described in this clause shall implement the virtual resource allocation pattern as defined in 5.2.2.

8.2.8.2 Flavors of allocation data

Various flavors of allocation data are defined:

- Processor resource allocation requests; see 8.1.4.1.
- Processor resource allocations; see 8.1.4.2.
- Settings that define the capabilities or mutability of managed resources. The Allocation Capabilities Profile described in clause 7 specifies a capabilities model that conveys information about the capabilities and the mutability of managed resources in terms of RASD instances.
- Parameters in operations that define or modify any of the representations listed above. The System Virtualization Profile described in clause 6 that specifies methods for the definition and modification of virtual resources. These methods use RASD instances for the parameterization of resource-allocation-specific properties.

Table 103 lists acronyms that are used in subclauses of 8.2.8 in order to designate RASD instances that represent various flavors of processor resource allocation data.

Table 103 – Acronyms for RASD adapted for the representation of various flavors of allocation data

Acronym	Flavor
Q_RASD	RASD adapted for the representation of processor resource allocation requests
R_RASD	RASD adapted for the representation of processor resource allocations
C_RASD	RASD adapted for the representation of settings that define capabilities of systems or processor resource pools, or that define the mutability of processor resource allocations or processor resource allocation requests
D_RASD	RASD adapted for the representation of new processor resource allocation requests in method parameter values
M_RASD	RASD adapted for the representation of modified processor resource allocations or processor resource allocation request in method parameter values

Subclauses of 8.2.8 detail implementation requirements for property values in RASD instances. In some cases requirements only apply to a subset of the flavors listed in Table 103; this is marked in the text through the use of respective acronyms.

8.2.8.3 CIM_ResourceAllocationSettingData class

This subclause defines rules for the values of properties in instances of the CIM_ResourceAllocationSettingData (RASD) class representing processor resource allocation

information representing the various flavors of processor resource allocation information defined in Table 103.

8.2.8.3.1 ResourceType property

The value of the ResourceType property in RASD instances representing processor resource allocation information shall be set to 3 (Processor) for processor resource allocation data.

8.2.8.3.2 PoolID property

The value of the PoolID property shall designate the processor resource pool. A NULL value shall indicate the use of the host system's default processor resource pool.

8.2.8.3.3 ConsumerVisibility property

The value of the ConsumerVisibility property shall denote whether host processor is directly passed through to the virtual system or whether processor is virtualized. Values shall be assigned as follows:

- A value of 2 (Passed-Through) shall denote that the virtual processor is based on a passed-through host processor.
- A value of 3 (Virtualized) shall denote that processor is virtualized.
- In the cases of { Q_RASD | D_RASD | M_RASD), a value of 0 (Unknown) shall indicate that the processor resource allocation request does not predefine which type of processor shall be allocated.

Other values shall not be used.

8.2.8.3.4 HostResource[ ] array property

The implementation of the HostResource[ ] array property is conditional.

Condition: The value 2 (Passed-Through) is supported for the value of the ConsumerVisibility property, or any of the values 3 (Dedicated), 4 (Soft Affinity) or 5 (Hard Affinity) is supported for the MappingBehavior property.

If HostResource[ ] array property is implemented, the provisions in this subclause apply.

In the cases of { Q_RASD | C_RASD | D_RASD | M_RASD } the value of the HostResource[ ] array property shall refer to (the representation of) one or more host resources that are configured to contribute processing power for the processor resource allocation.

In the case of R_RASD the value of the HostResource[ ] array property shall refer to (the representation of) the host resource(s) that contribute processing power for the processor resource allocation.

Elements of the value of the HostResource[ ] array property shall refer to instances of CIM classes, using the WBEM URI format as specified by DSP0207.

8.2.8.3.5 AllocationUnits property

The value of the AllocationUnits property shall be expressed in one of the following programmatic units: "hertz", "percent" or "count" or a multiple of the units expressed through a regular expression, as defined in DSP0004 programmatic units.

NOTE The units defined by value of the AllocationUnits property applies to the values of the Reserved and the Limit property; it does not apply to the value of the VirtualQuantity property.

8.2.8.3.6 VirtualQuantity property

The value of the VirtualQuantity property shall denote the number of virtual processors available to a virtual system.

NOTE The value of the VirtualQuantity property is a count; it is not expressed in allocation units. The units used for VirtualQuantity may be expressed in VirtualQuantityUnits (see 8.2.8.3.11).

8.2.8.3.7 Reservation property

The implementation of the Reservation property is optional.

If the Reservation property is implemented, the value of the Reservation property shall denote the minimum amount of host processing resources reserved for the use of a virtual system, expressed in allocation units.

8.2.8.3.8 Limit property

The implementation of the Limit property is optional.

If the Limit property is implemented, the value of the Limit property shall denote the maximum amount (or limit) of host processing power available to a virtual system, expressed in allocation units.

8.2.8.3.9 Weight property

The implementation of the Weight property is optional.

If the Weight property is implemented, its value shall denote the relative priority of a processor resource allocation in relation to other processor resource allocations from the same resource pool.

8.2.8.3.10 MappingBehavior property

The implementation of the MappingBehavior property is optional.

If the MappingBehavior property is implemented, its value shall denote how host resources referenced by elements in the value of HostResource[ ] array property relate to the processor resource allocation.

In R_RASD instances the following rules apply to the value of the MappingBehavior property:

- A value of 2 (Dedicated) shall indicate that the represented processor resource allocation is provided by host processor resources as referenced by the value of the HostResource[ ] array property that are exclusively dedicated to the virtual system.
- A value of 3 (Soft Affinity) or 4 (Hard Affinity) shall indicate that the represented processor resource allocation is provided using host processor resource as referenced by the value of the HostResource[ ] array property.
- Other values shall not be used.

In Q_RASD instances the following rules apply to the value of the MappingBehavior property:

- The special value NULL or a value of 0 (Unknown) shall indicate that the processor resource allocation request does not require specific host resources.
- A value of 2 (Dedicated) shall indicate that the processor resource allocation request shall be provided by exclusively dedicated host processor resources as specified through the value of the HostResource[ ] array property.
- A value of 3 (Soft Affinity) shall indicate that the processor resource allocation request shall preferably be provided by host processor resources as specified through the value of the HostResource[ ] array property, but that other resources may be used if the requested resources are not available.

- A value of 4 (Hard Affinity) shall indicate that the processor resource allocation request shall preferably be provided by host processor resources as specified through the value of the HostResource[ ] array property and that other resources shall not be used if the requested resources are not available.
- Other values shall not be used.

8.2.8.3.11 VirtualQuantityUnits property

VirtualQuantityUnits is an optional property that may be used to denote the units of the virtual device being allocated from the resource pool. The value of VirtualQuantityUnits shall be "count".

8.2.8.3.12 ResourceSubType property

The value of the ResourceSubType property shall designate a resource subtype. The format of the value shall be as follows: "<org-id>:<org-specific>". The <org-id> part shall identify the organization that defined the resource subtype value; the <org-specific> part shall uniquely identify a resource subtype within the set of subtype defined by the respective organization.

8.2.8.4 Instance requirements

This subclause details processor resource allocation related instance requirements.

8.2.8.4.1 Representation of resource allocation requests

Each processor resource allocation request shall be represented by a Q_RASD instance; the provisions of 8.5.11 apply.

8.2.8.4.2 Representation of resource allocations

Each processor resource allocation shall be represented by a R_RASD instance; the provisions of 8.5.11 apply.

The R_RASD instance shall be associated to the Q_RASD instance representing the corresponding resource allocation request (see 8.2.8.4.1) through an instance of the CIM_ElementSettingData association; the provisions of 8.5.10 apply.

The R_RASD instance shall be associated to the CIM_ResourcePool instance providing resources for the allocation (see 8.2.3.8) through an instance of the CIM_ResourceAllocationFromPool association; the provisions of 8.5.4 apply.

Implementations may represent a resource allocation request and the corresponding resource allocation by one RASD instance; in this case the association requirements of this subclause apply correspondingly. Note that association instances that refer to the RA_SASD instance are only existent while the resource is allocated.

8.2.8.4.3 Representation of resource allocation capabilities

The allocation capabilities of a system or a resource pool shall be represented by a CIM_AllocationCapabilities instance that is associated to the CIM_System instance representing the system or to the CIM_ResourcePool instance representing the resource pool through an instance of the CIM_ElementCapabilities association. (See the Allocation Capabilities Profile described in clause 7.)

The settings that define the allocation capabilities of a processor resource pool shall be represented by C_RASD instances; the provisions of 8.5.11 apply.

The processor allocation capabilities of a host system shall be a superset of the processor allocation capabilities of all processor resource pools that are hosted by the host system.

8.2.8.4.4 Representation of resource allocation mutability

The mutability of a resource allocation or resource allocation request shall be represented by a CIM_AllocationCapabilities instance that is associated to the RASD instance representing the resource allocation or resource allocation request through an instance of the CIM_ElementCapabilities association. (See the Allocation Capabilities Profile described in clause 7.)

The settings that define the allocation capabilities of a processor resource pool shall be represented by C_RASD instances; the provisions of 8.5.11 apply.

8.2.9 Virtual processor

A virtual processor shall be represented by exactly one CIM_Processor instance; the provisions of 8.5.8 apply. That instance shall be associated to all of the following instances:

- the CIM_ComputerSystem instance that represents the virtual system through an instance of the CIM_SystemDevice association; the provisions of 8.5.15 apply.
- the RASD instance that represents processor resource allocation through an instance of the CIM_SettingsDefineState association; the provisions of 8.5.13 apply.
- the CIM_ResourcePool instance that represents the processor resource pool providing the resource allocation through an instance of the CIM_ElementAllocatedFromPool association; the provisions of 8.5.2 apply.

Implementations may implement DSP1022 for virtual processors.

8.3 Methods

This subclause details the requirements for supporting intrinsic operations and extrinsic methods for the CIM elements defined by the profile described in this clause.

8.3.1 Profile conventions for operations

For each profile class (including associations), the implementation requirements for operations, including for those in the following default list, are specified in class-specific subclauses of this subclause.

The default list of operations for all classes is:

- GetInstance( )
- EnumerateInstances( )
- EnumerateInstanceNames( )

For classes that are referenced by an association, the default list also includes

- Associators( )
- AssociatorNames( )
- References( )
- ReferenceNames( )

The implementation requirements for intrinsic operations and extrinsic methods of classes listed in subclause 8.5, but not addressed by a separate subclause of this clause are specified by the "Methods" clauses of respective base profiles, namely the Resource Allocation Profile (see clause 5) and the Allocation Capabilities Profile described (see clause 7). These profiles are specialized by the profile described in this clause, and in these cases the profile described in this clause does not add method specifications beyond those defined in its base profiles.

8.3.2 CIM_Processor for host processors

All operations in the default list in 8.3.1 shall be implemented as defined in DSP0200. Note that related profiles may define additional requirements on operations for the profile class.

8.3.3 CIM_Processor for virtual processor

All operations in the default list in 8.3.1 shall be implemented as defined in DSP0200. Note that related profiles may define additional requirements on operations for the profile class.

8.3.4 CIM_ReferencedProfile

All operations in the default list in 8.3.1 shall be implemented as defined in DSP0200. Note that related profiles may define additional requirements on operations for the profile class.

8.3.5 CIM_RegisteredProfile

All operations in the default list in 8.3.1 shall be implemented as defined in DSP0200. Note that related profiles may define additional requirements on operations for the profile class.

8.3.6 CIM_SystemDevice for host processors

All operations in the default list in 8.3.1 shall be implemented as defined in DSP0200. Note that related profiles may define additional requirements on operations for the profile class.

8.3.7 CIM_SystemDevice for virtual processors

All operations in the default list in 8.3.1 shall be implemented as defined in DSP0200. Note that related profiles may define additional requirements on operations for the profile class.

8.4 Use cases

Clause 6.4 describes use cases for virtual system definition, modification, and destruction.

Clause 12.4 describes use cases for discovery of virtual systems, determination of the state and properties of a virtual system and the defined virtual system. This subclause is a pre-requisite to understanding the following use cases.

Clause 13.4 covers a number of essential use cases and should be read.

The following use cases and object diagrams illustrate use of the profile described in this clause. They are for informative purposes only and do not introduce behavioral requirements for implementations of the profile.

Figure 18 is a general instance diagram showing the essential classes for processor resource allocation.

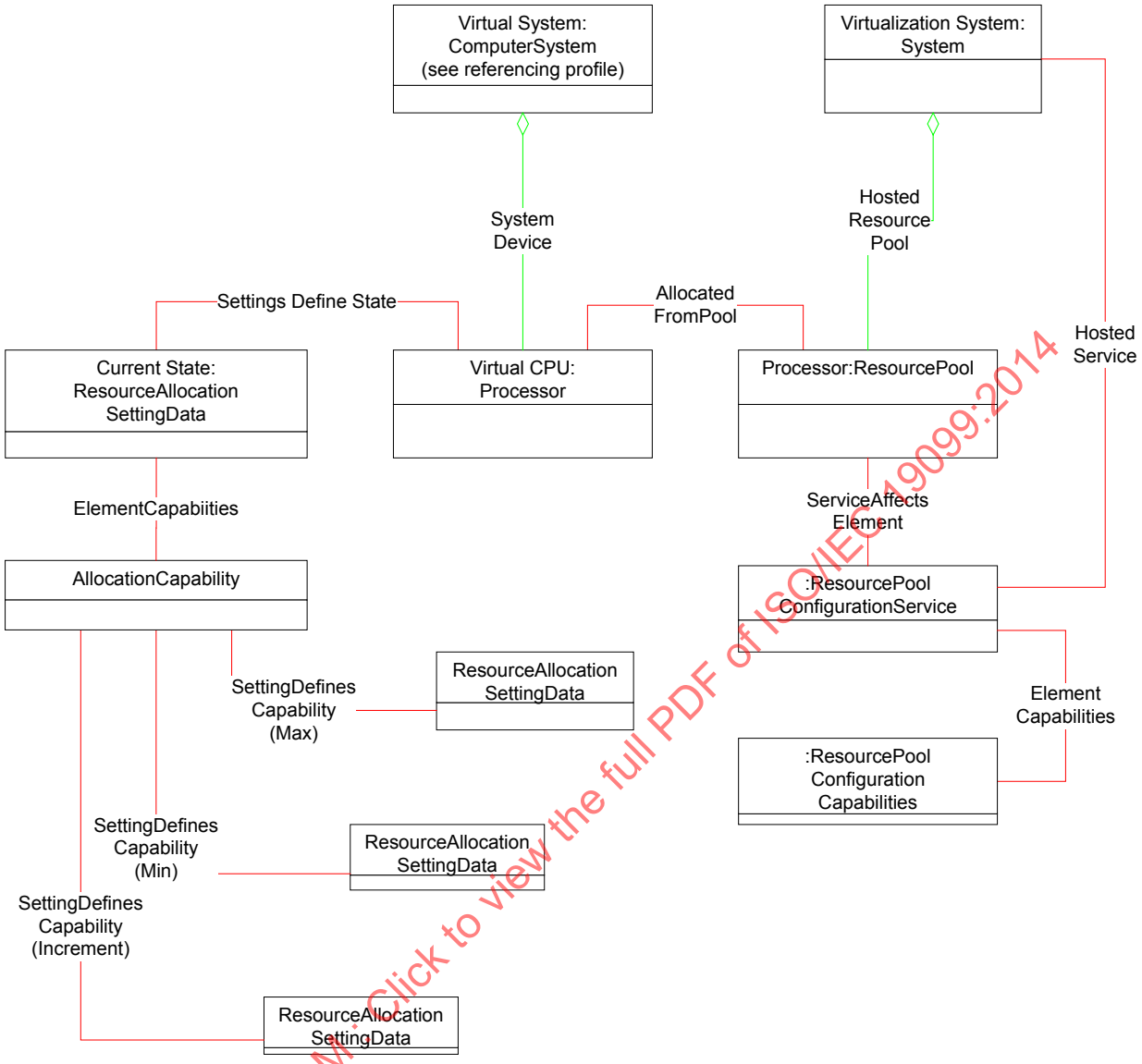

Figure 18 – Processor Resource Virtualization Profile: Instance diagram

There are different allocation units that may be used for the processor resource pool.

If instances of CIM_processor are instantiated to represent the physical processors aggregated into the primordial pool those instances should conform to DSP1022.

A defined state for allocation of processor resources is shown in Figure 19.

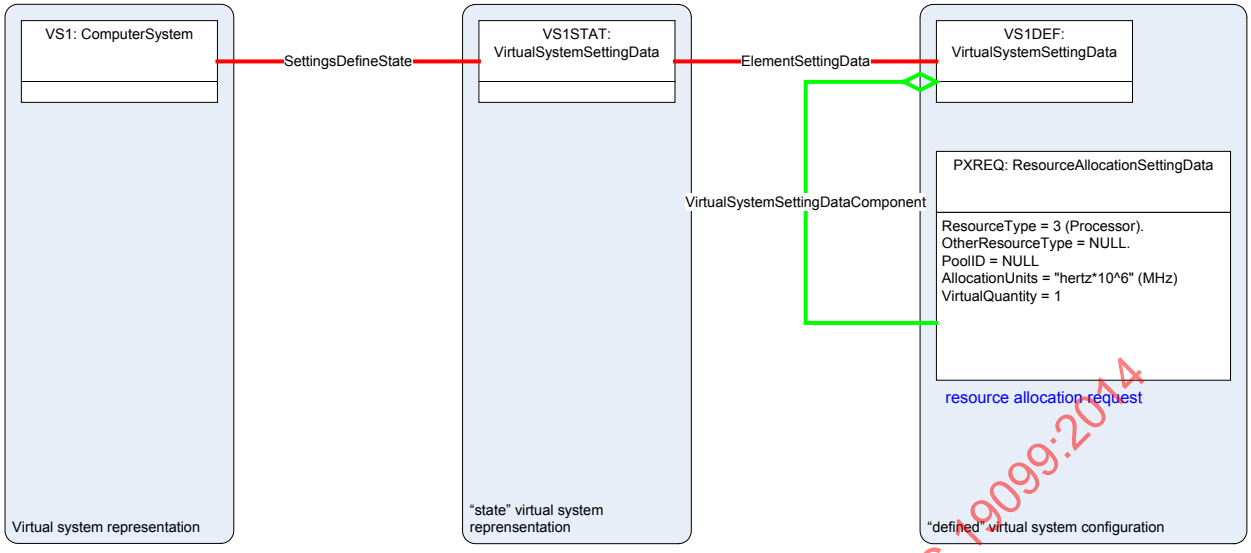

Figure 19 – Defined state

An active state for allocation of processor resources is shown in Figure 20.

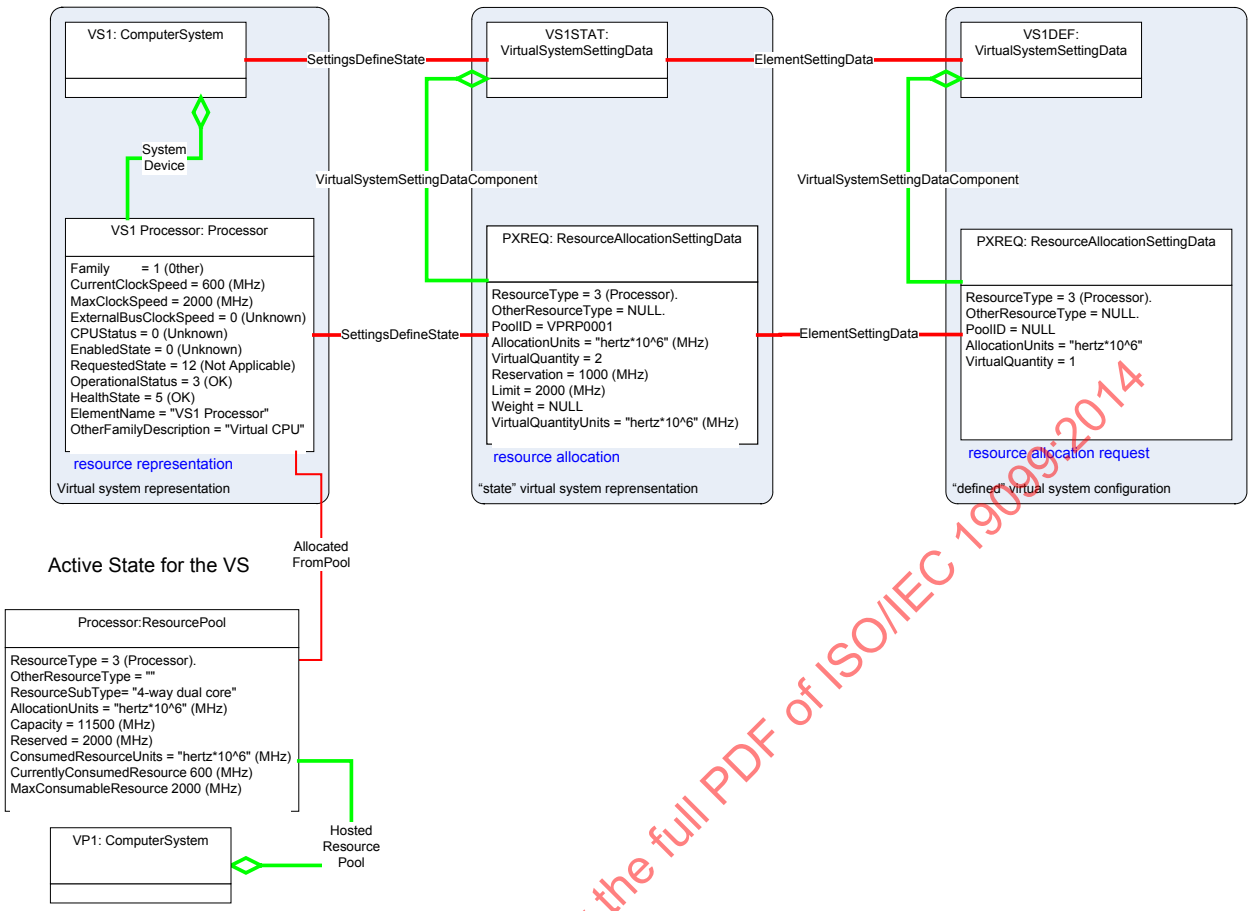

Figure 20 – Active state

8.4.1 Use case 1 – Increase the allocated processor capacity to a virtual machine in MHz.

8.4.1.1 Pre-conditions

The virtual system is active.

The CIM_ComputerSystem instance representing the virtual system is known.

The CIM_ResourcePool instance representing the processor resource pool is known.

The CIM_AllocationUnits of the resource pool is “hertz*10^6”.

The defined CIM_ResourceAllocationSettingData for the processor has property Reservation = 1000 (that is, 1GHz of processer capacity is allocated to the existing active virtual system) and has property Limit = 3000. The Reservation represents the minimum guaranteed resource available, even when system overcommitted. The sum of the Reservations must be less than the capacity of the resource pool. The Limit represents the maximum resource consumption, even when under committed.

8.4.1.2 Main

The client changes the defined CIM_ResourceAllocationSettingData for the processor property to Reservation = 2000.

The client invokes the CIM_VirtualManagementService.ModifyResourceSettings( ) method.

Check the return code value for success execution.

8.4.1.3 Post-conditions

The minimum processor capacity required to be allocated for the support of the virtual processor is now 2 GHz, the maximum admissible processor capacity is now 3 GHz.

8.4.2 Use case 2 – Increase the allocated processor capacity to a virtual machine in percent.

8.4.2.1 Pre-conditions

The virtual system is active.

The CIM_ComputerSystem instance representing the virtual system is known.

The CIM_ResourcePool instance representing the processor resource pool is known.

The CIM_ResourceAllocationSettingData.AllocationUnits of the resource pool is “%”.

The defined CIM_ResourceAllocationSettingData for the processor has property Reservation = 10 (that is, ten percent of processer capacity is allocated to the existing active virtual system) and has property Limit = 15.

8.4.2.2 Main

The client changes the defined CIM_ResourceAllocationSettingData for the processor property Reservation = 15.

The client invokes the CIM_VirtualSystemManagementService.ModifyResourceSettings( ) method.

Check the return code value for success execution.

8.4.2.3 Post-conditions

The processor resource allocated to the virtual system is now 15 percent minimum and the maximum processor resource that the virtual system can consume is 15 percent.

8.4.3 Use case 3 – Add a virtual processor to a virtual system

8.4.3.1 Pre-conditions

The virtual system is in-active.

The CIM_ComputerSystem instance representing the virtual system is known.

The CIM_ResourcePool instance representing the processor resource pool is known.

TheCIM_ResourceAllocationSettingData.AllocationUnits of the resource pool is “hertz*10^6”.

Figure 21 illustrates the instance diagram before the CIM_VirtualSystemManagementService.ModifyResourceSettings( ) method is called.

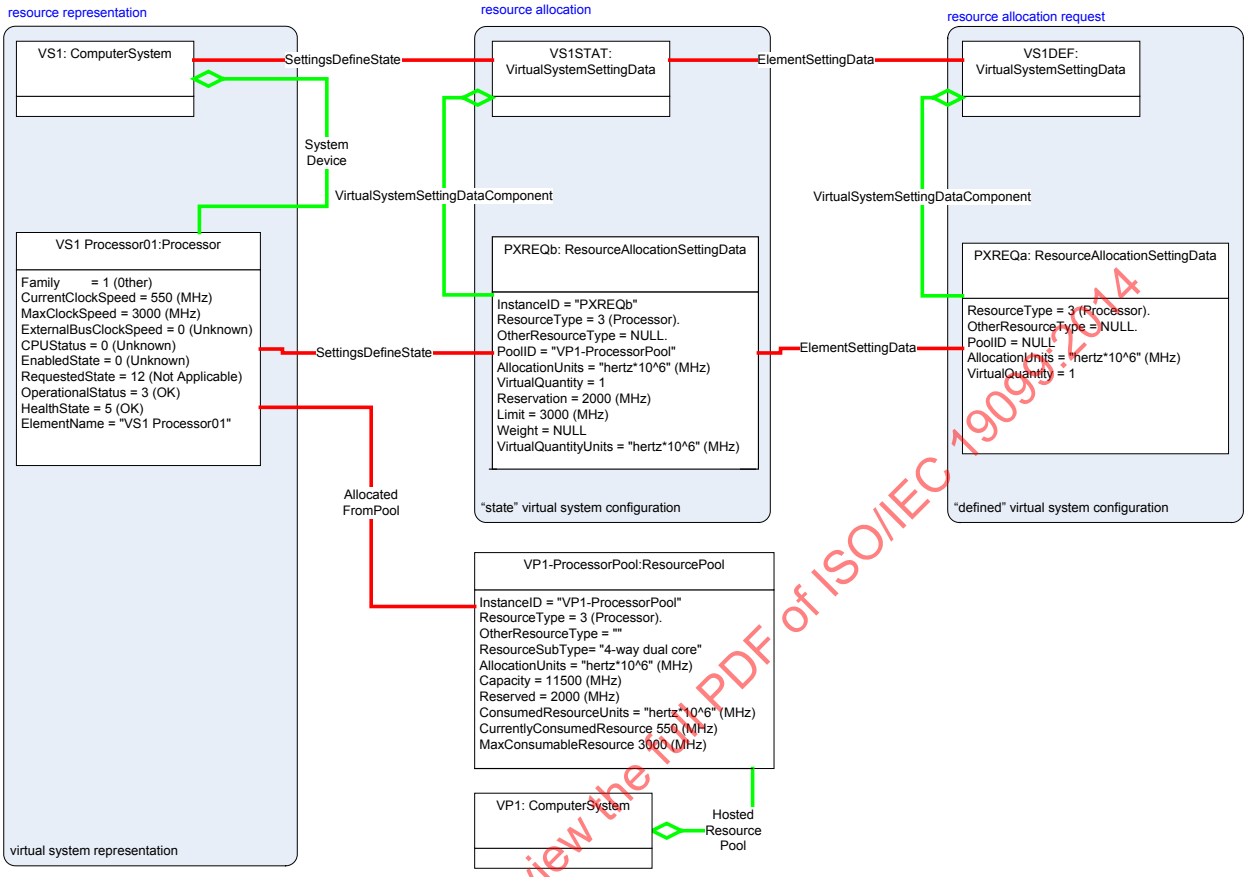

Figure 21 – CIM_ModifyResourceSettings – Before

Figure 22 illustrates the resource allocation setting data transferred with the CIM_VirtualSystemManagementService.ModifyResourceSettings( ) method call to increase the number of virtual processors.

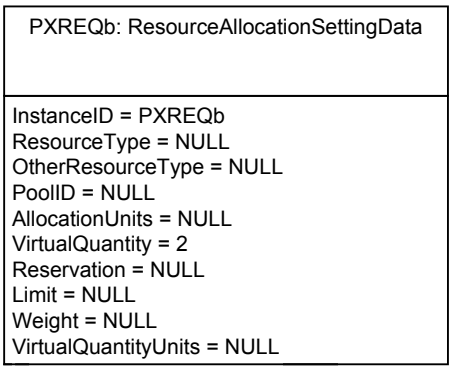

Figure 22 – RASD to Modify Resources

8.4.3.2 Main

Check the maximum value of the CIM_ResourceAllocationSettingData.VirtualQuantity to verify that a virtual processor addition is allowed.

Check the increment value of the CIM_ResourceAllocationSettingData.VirtualQuantity to verify that a one processor addition is allowed.

The client changes the CIM_ResourceAllocationSettingData.VirtualQuantity by one.

The client invokes the CIM_VirtualSystemManagement.Service.ModifyResourceSettings( ) method.

Check the return code value for success execution.

8.4.3.3 Post-conditions

The virtual quantity of processors allocated to the virtual system is now one greater.

Figure 23 illustrates the instance diagram after the CIM_VirtualSystemManagementService.ModifyResourceSettings( ) method is called.

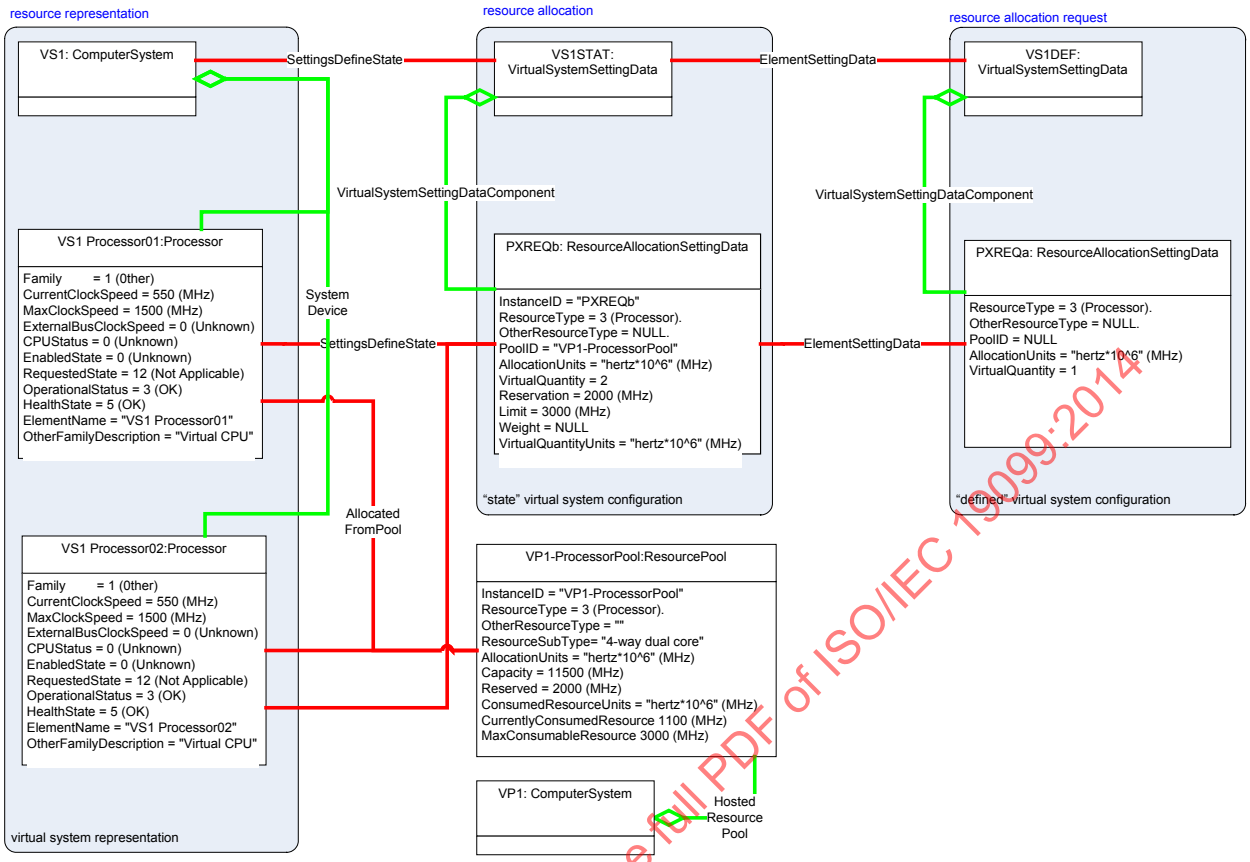

Figure 23 – CIM_ModifyResourceSettings – After

8.4.4 Use case 4 – Allocation of processor resource by weight

8.4.4.1 Pre-conditions

The virtual system is in-active.

The CIM_ComputerSystem instance representing the virtual system is known.

The CIM_ResourcePool instance representing the processor resource pool is known.

The CIM_AllocationUnits of the resource pool is “hertz*10^6”.

The defined CIM_ResourceAllocationSettingData for the processor has property Reservation = 0 (that is, no minimum level of processer capacity is allocated to the existing active virtual system) and has property Limit = none.

8.4.4.2 Main

The client changes the defined CIM_ResourceAllocationSettingData for the processor property Weight = 200.

The client invokes the CIM_VirtualSystemManagementService.ModifyResourceSettings( ) method.

Check the return code value for success execution.

The client activates the virtual system.

8.4.4.3 Post-conditions

The processor resource allocated to the virtual system is now 200/ SUM (Weight for each active virtual system). For example, if there are three other virtual systems with their Weight properties set to 200, 300, 100, then this virtual system is allocated 200/(200+300+100+200) or 25% of the processer capacity.

8.4.5 Use case 5 – Allocation of an additional processor resource

8.4.5.1 Pre-conditions

The virtual system is active.

The CIM_ComputerSystem instance representing the virtual system is known.

The CIM_ResourcePool instances representing the processor resource pools is known.

The CIM_AllocationUnits of the resource pools is “hertz 10^6”.

Figure 24 illustrates the instance diagram before the CIM_VirtualSystemManagementService.AddResourceSettings( ) method is called.

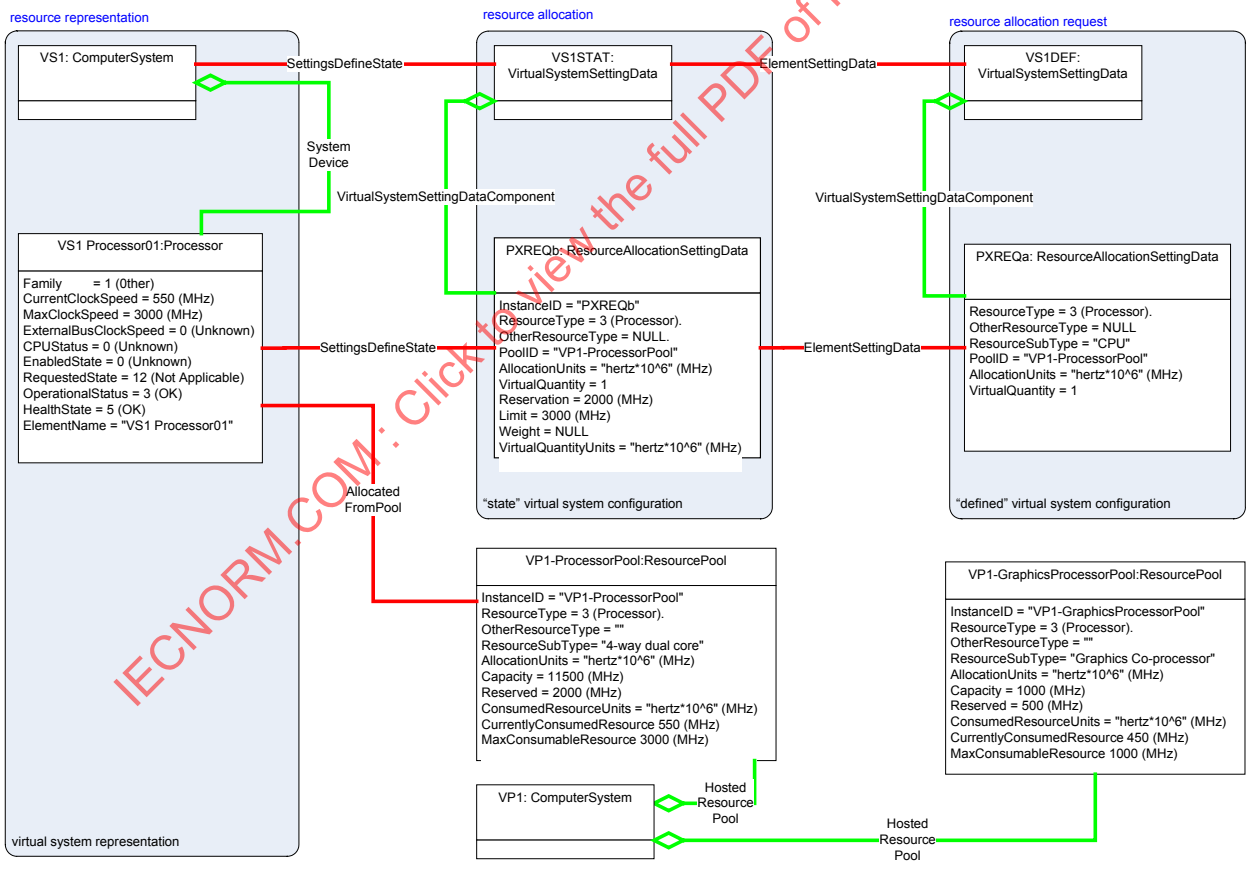

Figure 24 – CIM_AddResourceSettings – Before

Figure 25 illustrates the resource allocation setting data transferred with the CIM_VirtualSystemManagementService.AddResourceSettings( ) method call to add an additional virtual processor to the virtual system.

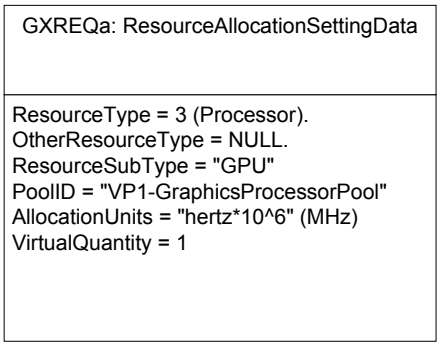

Figure 25 – RASD to add processor

8.4.5.2 Main

The client creates a defined CIM_ResourceAllocationSettingData specifying the PoolID for the resource pool from which the additional processor is to be allocated.

The client invokes the CIM_VirtualSystemManagementService.AddResourceSettings( ) method.

Check the return code value for success execution.

8.4.5.3 Post-conditions

The processor resources allocated to the virtual system is now two processors, one from the “VP1-ProcesssorPool” and one from the“VP1-GraphicsProcesssorPool”. Figure 26 illustrates the instance diagram after the CIM_VirtualSystemManagementService.AddResourceSettings( ) method is called.

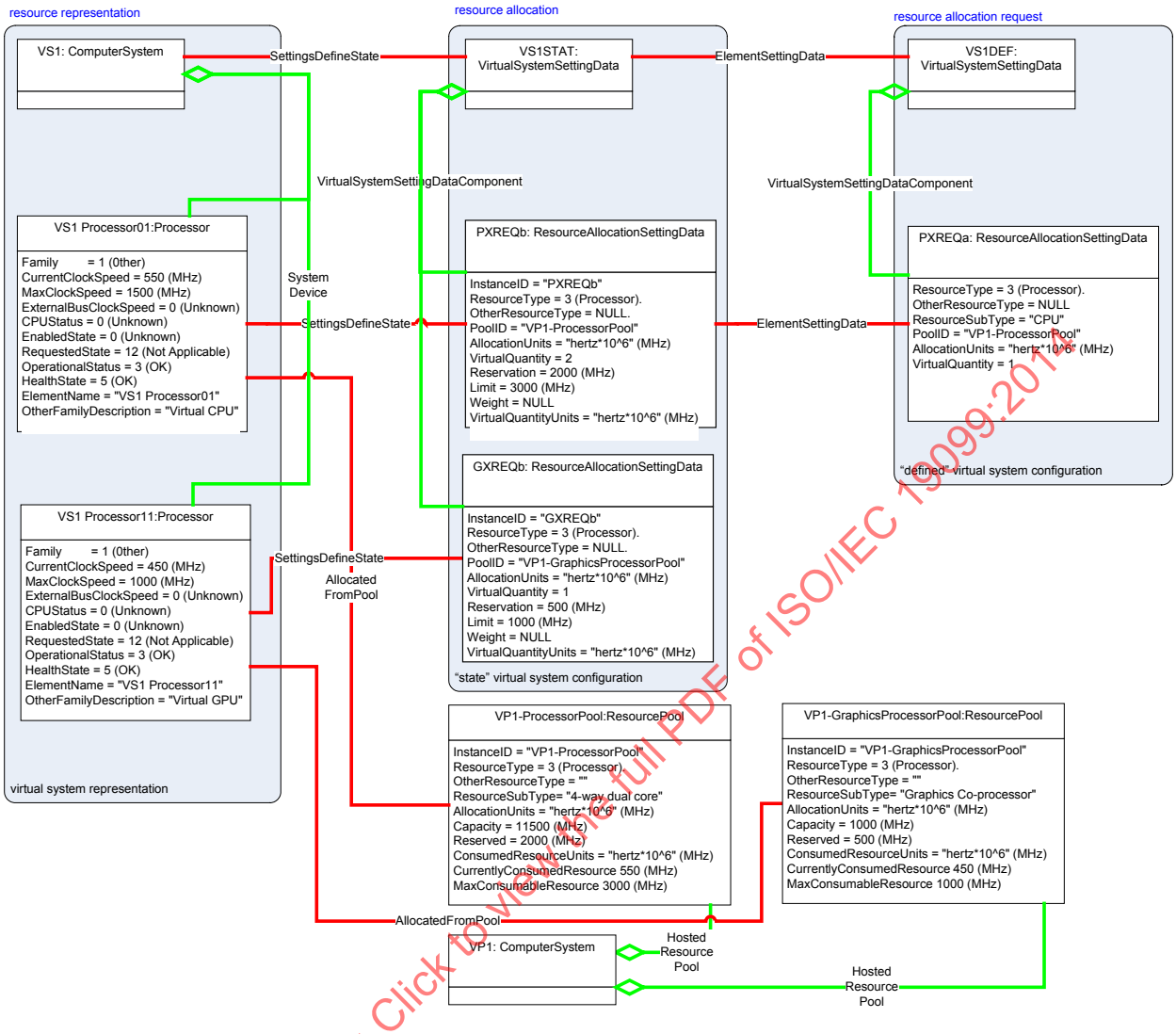

Figure 26 – CIM_AddResourceSettings – After

8.5 CIM elements

Table 104 lists CIM elements that are adapted by the profile described in this clause. Each CIM element shall be implemented as described in Table 104. The CIM Schema descriptions for any referenced element and its sub-elements apply.

Subclauses 8.2 ("Implementation") and 8.3 ("Methods") may impose additional requirements on these elements.

Table 104 – CIM Elements: Processor Resource Virtualization Profile

Element name	Requirement	Description
CIM_AffectedJobElement	Optional	See the Resource Allocation Profile described in clause 5.
CIM_AllocationCapabilities for capabilities	Mandatory	See the Allocation Capabilities Profile described in clause 7.
CIM_AllocationCapabilities for mutability	Optional	See the Allocation Capabilities Profile described in clause 7.
CIM_Component for resource pool	Conditional	See 8.5.1.
CIM_ConcreteJob	Optional	See the Resource Allocation Profile described in clause 5.
CIM_ElementAllocatedFromPool for allocated virtual processors	Mandatory	See 8.5.2.
CIM_ElementAllocatedFromPool for resource pool hierarchies	Conditional	See 8.5.3.
CIM_ElementCapabilities for capabilities	Mandatory	See the Allocation Capabilities Profile described in clause 7.
CIM_ElementCapabilities for mutability	Conditional	See the Allocation Capabilities Profile described in clause 7.
CIM_ElementCapabilities for resource pools	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_ElementSettingData for processor resource allocation	Mandatory	See 8.5.4.
CIM_ElementSettingData for processor resource pools	Conditional	See 8.5.5.
CIM_HostedDependency	Optional	See 8.5.6.
CIM_HostedResourcePool	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_HostedService	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_Processor for host processors	Conditional	See 8.5.7.
CIM_Processor for virtual processors	Mandatory	See 8.5.8.
CIM_ReferencedProfile	Mandatory	See the Virtual System Profile described in clause 12.
CIM_RegisteredProfile	Mandatory	See 8.5.9.
CIM_ResourceAllocationFromPool	Optional	See 8.5.10.
CIM_ResourceAllocationSettingData	Mandatory	See 8.5.11.
CIM_ResourcePool	Mandatory	See 8.5.12.
CIM_ResourcePoolConfigurationCapabilities	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_ResourcePoolConfigurationService	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_SettingsDefineState	Mandatory	See 8.5.13.

Element name	Requirement	Description
CIM_ServiceAffectsElement	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_SystemDevice for host processors	Conditional	See 8.5.14.
CIM_SystemDevice for virtual processors	Mandatory	See 8.5.15.
Indications		
None defined		

8.5.1 CIM_Component for resource pool

The implementation of the CIM_Component association for the representation of the aggregation of host resources into resource pools is conditional.

Condition: The representation of resource aggregation (see 8.2.4) is implemented.

The CIM_Component association is abstract; therefore it cannot be directly implemented. For this reason the provisions in this subclause shall be applied to implementations of subclasses of the CIM_Component association. However, note that clients may directly resolve abstract associations without knowledge of the concrete subclass that is implemented.

Table 105 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 105 – Association: CIM_Component for resource pool

Elements	Requirement	Notes
GroupComponent	Mandatory	**Key:** Value shall reference the CIM_ResourcePool instance that represents the resource pool. **Cardinality:** 0..1
PartComponent	Mandatory	**Key:** Value shall reference the CIM_ManagedElement instance that represents a component of the resource pool. **Cardinality:** *

8.5.2 CIM_ElementAllocatedFromPool for allocated virtual processors

Table 106 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 106 – Association: CIM_ElementAllocatedFromPool

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Value shall reference the instance of the CIM_ResourcePool class that represents a processor resource pool. **Cardinality:** 1
Dependent	Mandatory	**Key:** Value shall reference the instance of the CIM_Processor class that represents virtual processor resulting from a processor allocation from the pool. **Cardinality:** *

8.5.3 CIM_ElementAllocatedFromPool for resource pool hierarchies

The implementation of the CIM_ElementAllocatedFromPool association for the representation of resource pool hierarchies is conditional.

Condition: Resource pool management (see 8.2.7) is implemented.

Table 107 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 107 – Association: CIM_ElementSettingData

Elements	**Requirement**	**Notes**
Antecedent	Mandatory	**Key:** Value shall reference the CIM_ResourcePool instance that represents the parent resource pool. **Cardinality:** 1
Dependent	Mandatory	**Key:** Value shall reference the CIM_ResourcePool instance that represents the child resource pool. **Cardinality:** *

8.5.4 CIM_ElementSettingData for processor resource allocation

Table 108 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 108 – Association: CIM_ElementSettingData for processor resource allocation

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Value shall reference the RASD instance represents the processor resource allocation. **Cardinality:** 1
SettingData	Mandatory	**Key:** Value shall reference the RASD instance that represents the processor resource allocation request. **Cardinality:** 0..1
IsDefault	Mandatory	Value shall be 1 (Is Default).

8.5.5 CIM_ElementSettingData for processor resource pool

The implementation of the CIM_ElementSettingData class for the representation of the relationship between a child resource pool and its processor resource allocation is conditional.

Condition: Processor resource pool hierarchies (see 8.2.5) are implemented.

Table 109 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 109 – Association: CIM_ElementSettingData (Processor Resource Pool)

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Value shall reference the CIM_ResourcePool instance that represents the concrete processor resource pool. **Cardinality:** 0..1
SettingData	Mandatory	**Key:** Value shall reference the RASD instance that represents the processor resource allocation. **Cardinality:** 0..1

8.5.6 CIM_HostedDependency

The support of the CIM_HostedDependency association is optional.

Table 110 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 110 – Association: CIM_HostedDependency

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Value shall reference the CIM_Processor instance that represents host processor. **Cardinality:** 0..1
Dependent	Mandatory	**Key:** Value shall reference the CIM_Processor instance that represents virtual processor. **Cardinality:** 0..1

8.5.7 CIM_Processor (host processor)

The implementation of the CIM_Processor class for the representation of host processors is conditional.

Condition: The representation of host resources is implemented; see 8.2.2.

Table 111 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in DSP1022 if that is implemented.

Table 111 – Class: CIM_Processor (host processor)

Elements	Requirement	Notes
SystemCreationClassName	Mandatory	**Key**
CreationClassName	Mandatory	**Key**
SystemName	Mandatory	**Key**
Name	Mandatory	**Key**
EnabledState	Mandatory	See CIM schema description.
RequestedState	Mandatory	See CIM schema description.

8.5.8 CIM_Processor (virtual processor)

See 8.2.9 for detailed implementation requirements for this class adaptation.

Table 112 lists the requirements for elements of this class adaptation. These requirements are in addition to those specified in the CIM Schema and in DSP1022 if that is implemented.

Table 112 – Class: CIM_Processor (virtual system)

Elements	Requirement	Notes
SystemCreationClassName	Mandatory	**Key**
CreationClassName	Mandatory	**Key**
SystemName	Mandatory	**Key**
Name	Mandatory	**Key**
EnabledState	Mandatory	Unspecified.
RequestedState	Mandatory	Unspecified.

8.5.9 CIM_RegisteredProfile

The basic adaptation of the CIM_RegisteredProfile class is specified by DSP1033.

Table 113 lists the requirements for elements of this class. These requirements are in addition to those specified in DSP1033.

Table 113 – Class: CIM_RegisteredProfile

Elements	Requirement	Notes
RegisteredOrganization	Mandatory	Value shall be set to 2 (DMTF).
RegisteredName	Mandatory	Value shall be set to “Processor Resource Virtualization”.
RegisteredVersion	Mandatory	Value shall be set to the version of the profile described in this clause: “1.0.0”.

8.5.10 CIM_ResourceAllocationFromPool

The support of the CIM_ResourceAllocationFromPool association is optional.

Table 114 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 114 – Association: CIM_ResourceAllocationFromPool

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Value shall reference the instance of the CIM_ResourcePool class that represents a processor resource pool. **Cardinality:** 0..1
Dependent	Mandatory	**Key:** Value shall reference the instance of the CIM_ResourceAllocationSettingData class that represents a processor resource allocation from the pool. **Cardinality:** *

8.5.11 CIM_ResourceAllocationSettingData

See 8.2.8.3 for detailed implementation requirements for this class.

Table 115 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 115 – Class: CIM_ResourceAllocationSettingData

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**; see the Resource Allocation Profile described in clause 5.
ResourceType	Mandatory	Value shall be 3 (Processor).
OtherResourceType	Mandatory	Value shall be NULL.
ResourceSubType	Optional	See 8.2.8.3.12.
PoolID	Mandatory	See 8.2.8.3.2.
ConsumerVisibility	Optional	See 8.2.8.3.3.
HostResource[ ]	Optional	See 8.2.8.3.4.
AllocationUnits	Mandatory	See 8.2.8.3.5.
VirtualQuantity	Mandatory	See 8.2.8.3.6.
Reservation	Optional	See 8.2.8.3.7.
Limit	Optional	See 8.2.8.3.8.
Weight	Optional	See 8.2.8.3.9.
AutomaticAllocation	Optional	See the Resource Allocation Profile described in clause 5.
AutomaticDeallocation	Optional	See the Resource Allocation Profile described in clause 5.
MappingBehavior	Optional	See 8.2.8.3.10.
VirtualQuantityUnits	Optional	See 8.2.8.3.11

8.5.12 CIM_ResourcePool

Table 116 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 116 – Class: CIM_ResourcePool

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
ElementName	Optional	See the Resource Allocation Profile described in clause 5.
PoolID	Mandatory	See 8.2.3.3.
Primordial	Mandatory	See 8.2.3.4.
Capacity	Conditional	See 8.2.3.6.
Reserved	Optional	See 8.2.3.5.
ResourceType	Mandatory	Value shall be 3 (Processor).
OtherResourceType	Mandatory	Value shall be NULL.
ResourceSubType	Optional	See 8.2.3.2.
AllocationUnits	Mandatory	See 8.2.1

8.5.13 CIM_SettingsDefineState

Table 117 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 117 – Association: CIM_SettingsDefineState

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Value shall reference an instance of the CIM_Processor class. **Cardinality:** 0..1
SettingData	Mandatory	**Key:** Value shall reference the instance of the CIM_ResourceAllocationSettingData class. **Cardinality:** 0..1

8.5.14 CIM_SystemDevice for host processor

The implementation of the CIM_SystemDevice association for the representation of the relationship between host processors and their system is conditional.

Condition: The representation of host resources is implemented (see 8.2.2).

NOTE If DSP1022 is implemented for host processors, the implementation of the CIM_SystemDevice association for the representation of the relationship between host processors and their system is required.

If the CIM_SystemDevice association is implemented for the representation of the relationship between host processors and their systems, the provisions in this subclause apply.

Table 118 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema, in the Resource Allocation Profile described in clause 5, and in DSP1022 if that is implemented.

Table 118 – Association: CIM_SystemDevice (Host Processor)

Elements	Requirement	Notes
GroupComponent	Mandatory	**Key:** Value shall reference an instance of the CIM_System class. **Cardinality:** 1
PartComponent	Mandatory	**Key:** Value shall reference the instance of the CIM_Processor class. **Cardinality:** *

8.5.15 CIM_SystemDevice for virtual processor

Table 119 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 119 – Association: CIM_SystemDevice (Virtual Processor)

Elements	Requirement	Notes
GroupComponent	Mandatory	**Key:** Value shall reference an instance of the CIM_System class. **Cardinality:** 1
PartComponent	Mandatory	**Key:** Value shall reference the instance of the CIM_Processor class. **Cardinality:** *

9 Memory Resource Virtualization Profile

Profile Name: Memory Resource Virtualization

Version: 1.0.0

Organization: DMTF

CIM Schema Version: 2.22

Central Class: CIM_ResourcePool

Scoping Class: CIM_System

The Memory Resource Virtualization Profile is a component profile that defines the minimum object model needed to provide for the CIM representation and management of the virtualization of memory.

Table 120 lists DMTF management profiles on which the profile described in this clause depends.

Table 120 – Related profiles for the Memory Resource Virtualization Profile

Profile name	Organization	Version	Relationship	Description
Resource Allocation	DMTF	1.1	Specializes	The abstract profile that describes the virtualization of resources (see clause 5) See 9.2.2.
Allocation Capabilities	DMTF	1.0	Specializes	The abstract profile that describes capabilities for resource allocation (see clause 7) See 9.2.3.
Profile Registration	DMTF	1.0	Mandatory	The profile that specifies registered profiles
System Memory	DMTF	1.0	Optional	The profile that specifies the management of system memory See 9.2.4.

9.1 Description (informative)

This subclause introduces the management domain addressed by the Memory Resource Virtualization Profile, and outlines the central modeling elements established for representation and control of the management domain.

9.1.1 General

In computer virtualization systems, virtual computer systems are composed of component virtual resources. The profile described in this clause specifies the allocation and management of host computer system memory in support of virtual computer system memory. The memory described here would appear as “physical” memory to an operating system running in the virtual computer system. The profile described in this clause is not intended to specify the management of virtual memory as virtualized by operating systems.

The profile described in this clause applies the resource virtualization pattern defined in the Resource Allocation Profile (see clause 5) and the allocation capabilities pattern defined in the Allocation

Capabilities Profile described in clause 7 to enable the management of processor resources that are allocated to virtual systems. The profile described in this clause defines additional CIM elements and constraints beyond those defined in the specialized profiles. Optionally implementations may implement DSP1026 (*System Memory Profile*) to represent host memory.

9.1.2 Memory resource virtualization class schema

Figure 27 shows the class schema of the profile described in this clause. It outlines the elements that are referenced and in some cases further constrained by the profile described in this clause, as well as the dependency relationships between elements of the profile described in this clause and other profiles. For simplicity in diagrams, the prefix *CIM_* has been removed from class and association names. Inheritance relationships are shown only to the extent required in the context of the profile described in this clause.

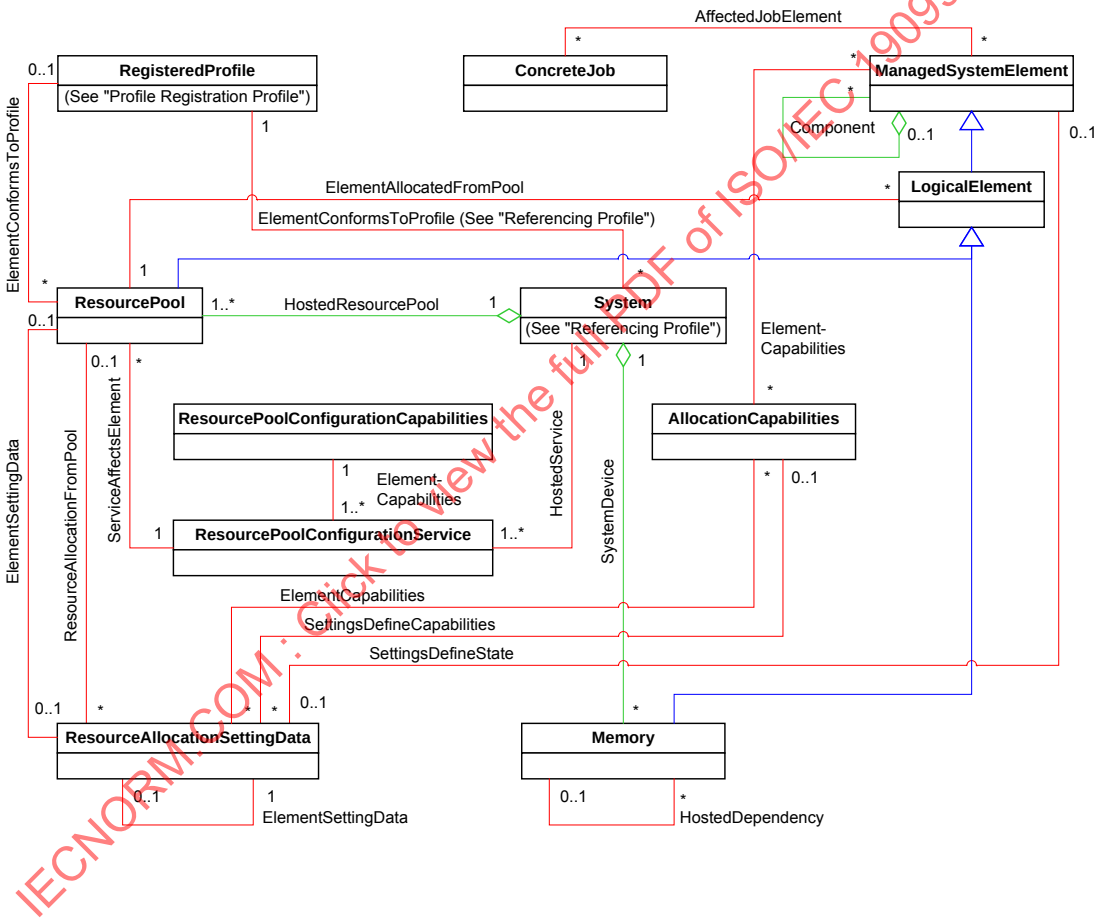

Figure 27 – Memory Resource Virtualization Profile: Profile class diagram

The profile described in this clause specifies the use of the following classes and associations:

- The CIM_ResourcePool class modeling resource pools for memory resources. Host memory resources are allocated from their resource pool and used to create the memory resources for virtual systems

- The CIM_Component association modeling the following relationships:
  - the relationship between memory resource pools and memory extents as components of the resource pools
  - the relationship between one aggregated memory extent and one or more aggregating memory extents
- The CIM_ElementAllocatedFromPool association modeling hierarchies of memory resource pools and modeling the relationship of resource pools and the virtual memory allocated from those
- The CIM_HostedResourcePool association modeling the hosting dependency between a memory resource pool and its host system. A host system supports at least one resource pool
- The CIM_Memory class modeling the following aspects of memory:
  - memory as a device in the scope of a system, as modeled by the CIM_SystemDevice association
  - memory as a result of a memory resource allocation from a resource pool, as modeled by the CIM_ElementAllocatedFromPool association
  - memory as a component within memory resource pools, as modeled through the CIM_Component association
- The CIM_ResourceAllocationSettingData class representing memory resource allocations or memory resource allocation requests
- The CIM_AllocationCapabilities class and the CIM_ElementCapabilities association modeling
  - the memory resource allocation capabilities of host systems
  - the memory resource allocation capabilities of memory resource pools
  - the mutability of existing memory allocations
- The CIM_SettingsDefineCapabilities association modeling the relation between memory allocation capabilities and the settings that define these capabilities
- The CIM_ResourcePoolConfigurationService modeling configuration services for memory resource pools and the CIM_ResourcePoolConfigurationCapabilities class modeling their capabilities
- The CIM_ConcreteJob class and the CIM_AffectedJobElement association modeling asynchronous management tasks initiated through memory resource pool configuration services

In general, any mention of a class in this document means the class itself or its subclasses. For example, a statement such as “an instance of the CIM_StorageExtent class” implies an instance of the CIM_StorageExtent class or a subclass of the CIM_StorageExtent class.

9.1.3 Memory resource pool

The profile described in this clause applies the concept of resource pools defined in the Resource Allocation Profile described in clause 5 to the memory resource type. The Memory Resource Virtualization Profile uses the memory resource pool as the focal point for memory allocations. Virtual systems receive memory allocations from memory resource pools based on memory resource allocation requests. A memory resource pool is an aggregation of host memory available for allocation in support of virtual memory.

Two types of memory resource pools are defined: primordial and concrete.

9.1.3.1 Primordial Memory Resource Pool

A primordial memory resource pool aggregates memory capacity; it represents a subset of the manageable memory capacity of a host system.

9.1.3.2 Concrete memory resource pool

A concrete memory resource pool subdivides the memory capacity of its parent resource pool. The amount of memory allocated to a concrete resource pool is less than or at most equal to the capacity of the parent pool, but may use all of the capacity of the parent pool.

9.1.3.3 Hierarchies of memory resource pools

The profile described in this clause applies the concept of resource pool hierarchies defined in the Resource Allocation Profile described in clause 5 to the memory resource type; see 5.1.1.4.

Figure 28 shows an example of the CIM representation of a memory resource pool hierarchy in which two host memory extents are aggregated into a primordial memory resource pool. The primordial memory resource pool is subdivided into three concrete memory resource pools, with each concrete resource pool allocated 1 GB of memory. The assigned weights differ for each concrete memory resource pool, indicating different qualities of service for memory extents that are allocated from these pools.

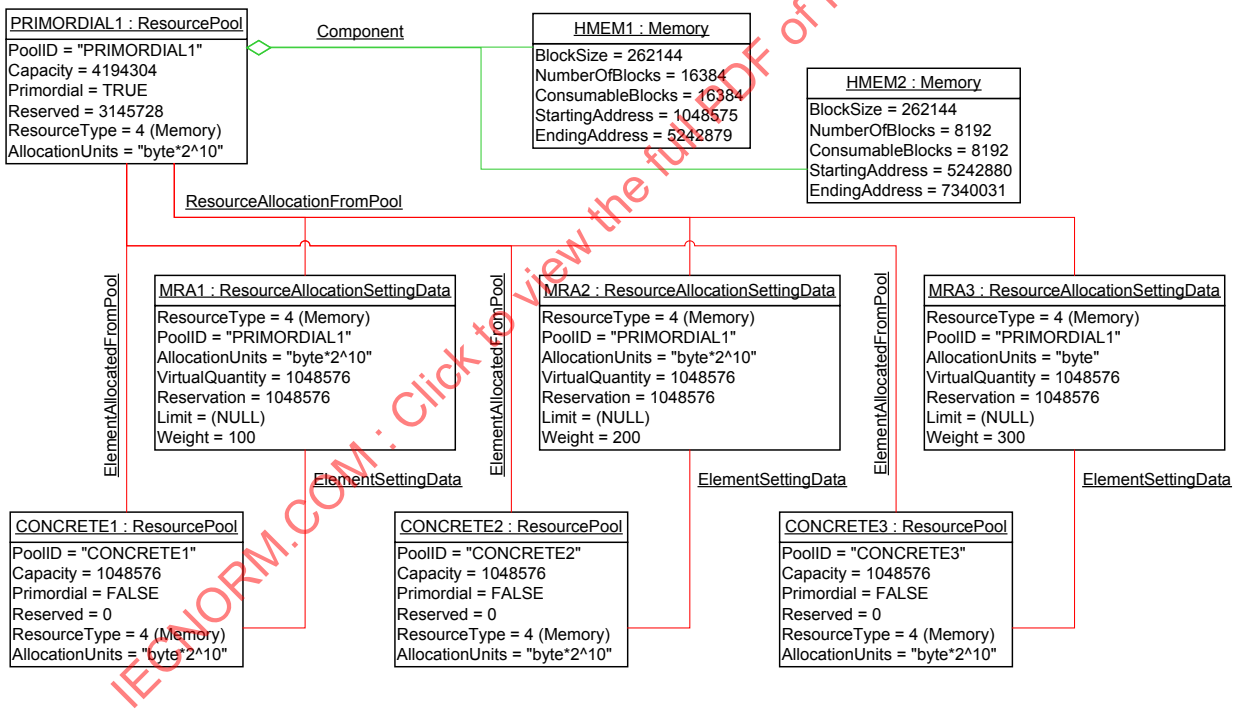

Figure 28 – Instance Diagram: Concept of memory resource pool hierarchies

9.1.3.4 Memory resource pool management

The profile described in this clause applies the concept of resource pool management defined in the Resource Allocation Profile described in clause 5 to the memory resource type; see 5.1.1.5.

9.1.4 Memory resource allocation

The profile described in this clause applies the concept of device resource allocation defined in the Resource Allocation Profile described in clause 5 to the memory resource type; see 5.1.3.

9.1.4.1 General

Figure 29 shows a typical situation for memory resource allocation in the context of a virtual system.

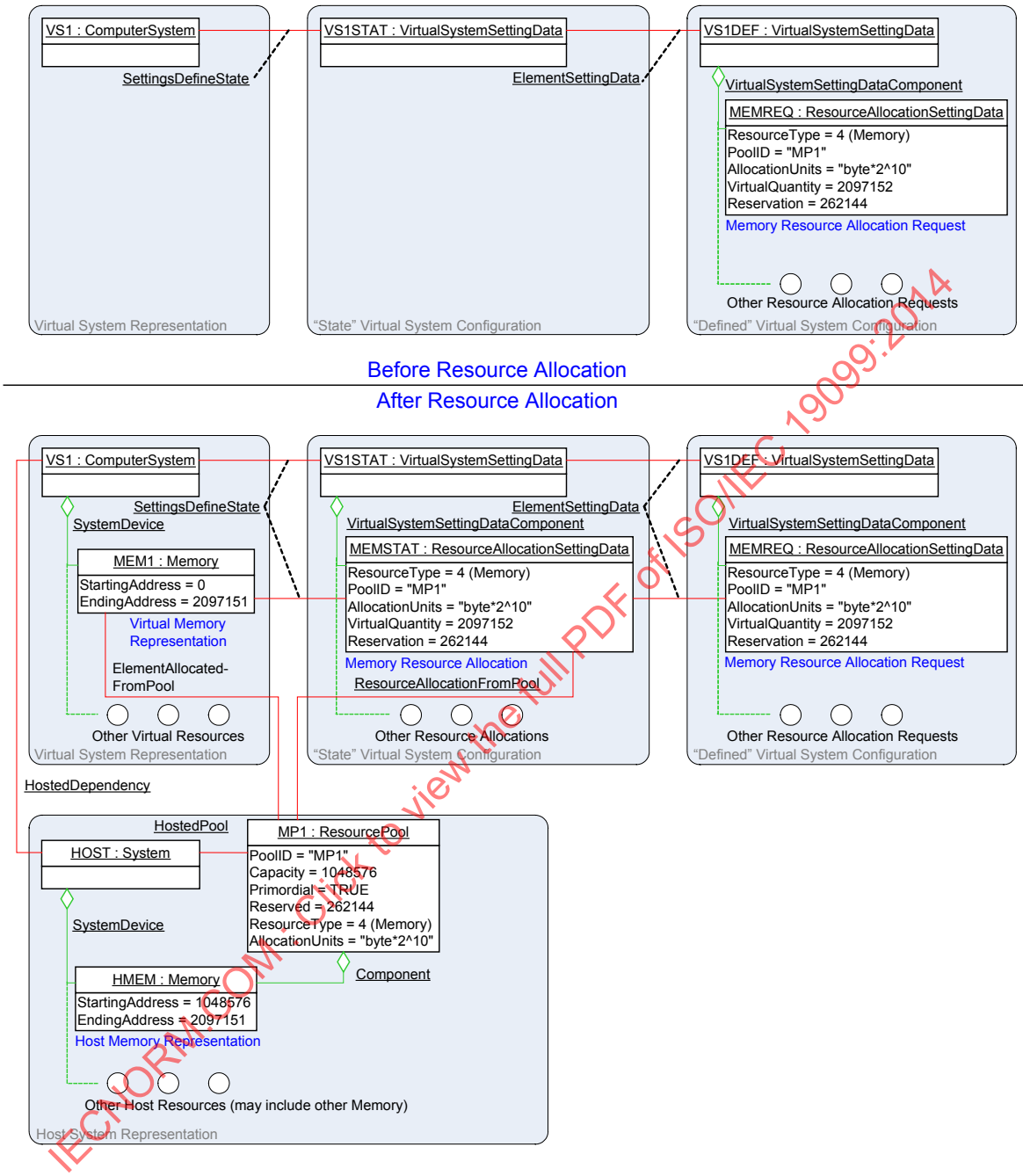

Figure 29 – Instance Diagram: Concept of memory resource allocation

9.1.4.2 Memory resource allocation request

The memory requirements of a virtual system are defined as part of the "Defined" virtual system configuration. The "Defined" virtual system configuration contains memory resource allocation requests represented as instances of the CIM_ResourceAllocationSettingData class.

An example of the CIM representation of a memory resource allocation request is shown in the upper right part of Figure 29.

9.1.4.3 Memory resource allocation

As a virtual system is activated (or instantiated), memory needs to be allocated as requested by memory resource allocation requests in the virtual system definition. Memory resource allocations are represented as instances of the CIM_ResourceAllocationSettingData class in the “State” virtual system configuration.

An example of the CIM representation of a memory resource allocation is shown in the center part of Figure 29.

9.1.4.4 Virtual memory

Virtual memory is the instantiation of allocated host memory that is exposed to a virtual system through a logical memory device; it is the result of the memory resource allocation based on a memory resource allocation request. Virtual memory may be virtualized using techniques like paging and dynamic address translation, but may also be host memory that is directly passed through to the virtual system. Virtual memory is represented by an instance of the CIM_Memory class as part of the virtual system representation.

NOTE The definition of the term “virtual memory” is specialized from the term “virtual resource” defined in the Resource Allocation Profile described in clause 5 and deviates from common computer industry parlance.

An example of the CIM representation of virtual memory as the result of a memory resource allocation is shown on the left side in the central part of Figure 29.

9.1.4.5 Memory virtualization

The amount of host memory reserved may be less than the amount of virtual memory available to the virtual system. This indicates that memory is virtualized by facilities of the host system, such that the amount of virtual memory usable by the virtual system is larger than the amount of real memory required for its support.

An example of the CIM representation of memory virtualization is shown in the center part of Figure 29 where the amount of virtual memory is significantly larger than the amount of allocated host memory.

9.1.4.6 Memory composition

Memory may be composed of other memory. For example, a virtualization platform may support the allocation of memory from more than one resource pool, resulting in several virtual memory extents that then are composed into one aggregating memory extent. This situation is shown in Figure 30. It is similar to the situation shown in Figure 29, but in Figure 30 the virtual memory is composed from two memory extents that are allocated from two different memory resource pools. The two memory extents are composed into a memory composition that is a logical device of the virtual system.

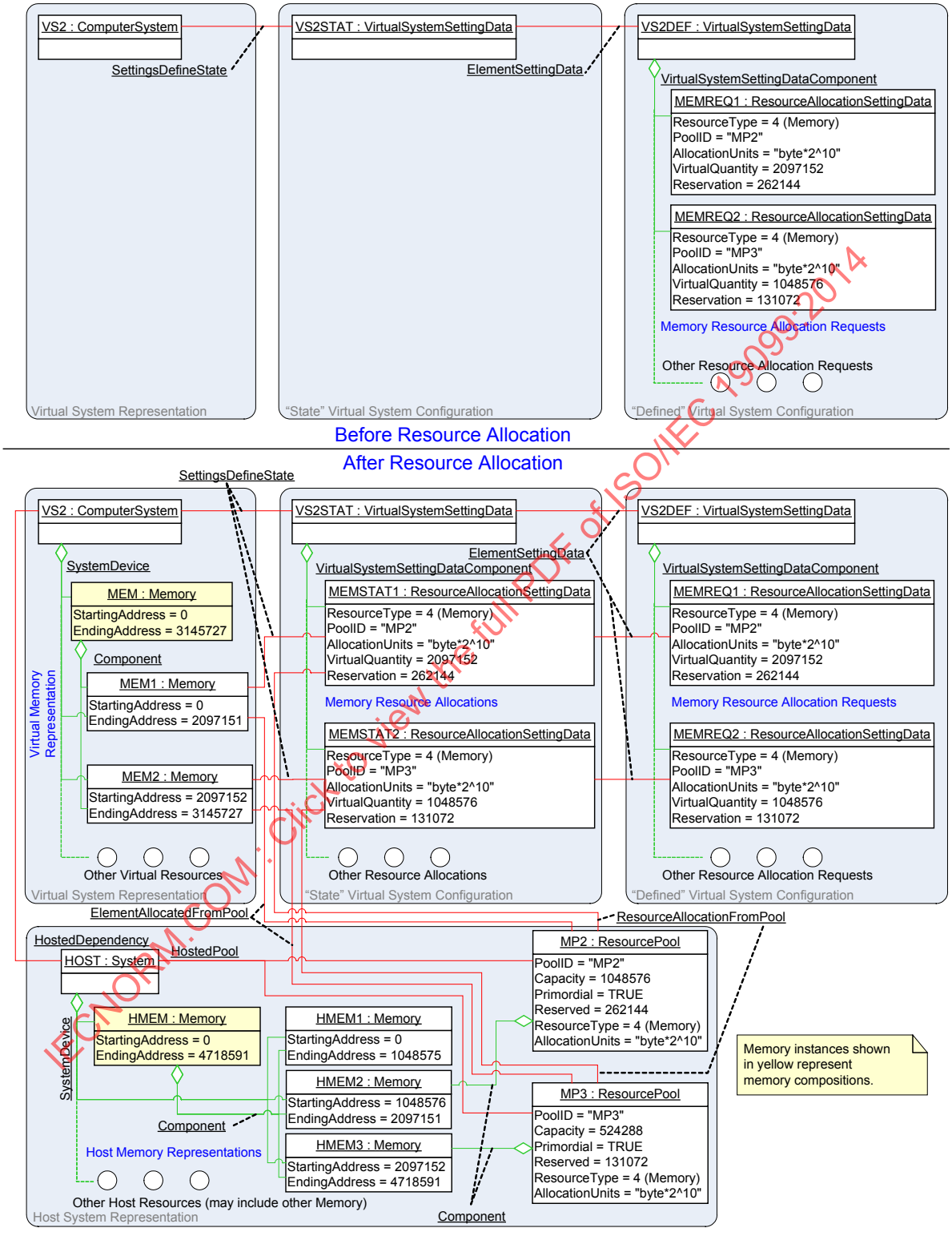

Figure 30 – Instance Diagram: Memory composition

9.1.4.7 Dedicated host memory

Dedicated host memory is memory owned by the host system that is exclusively reserved for support of the virtual memory of a particular virtual system.

9.1.4.8 Host memory consumption

A memory resource allocation request references the memory resource pool to be used by specifying the value of the PoolID property. Host memory is allocated from the identified memory resource pool during memory resource allocation.

9.1.4.8.1 Statically controlled memory consumption

With statically controlled memory consumption, a particular memory resource allocation request is supported only if the amount of host memory available in the addressed memory pool is at least as large as the amount requested. This approach is called admission control. Each successfully allocated memory resource reduces the amount of memory available in the pool, respectively. In the CIM representation of the memory resource pool, the amount of memory consumed from the pool is visible through the value of the Reserved property.

9.1.4.8.2 Dynamically controlled memory consumption

With dynamically controlled memory consumption, the amount of host memory allocated in support of virtual memory is not a constant value but varies significantly over time, depending on factors like the memory access pattern of software executed within the virtual system or the memory consumption of other virtual machines.

Further, with dynamically controlled memory consumption, the amount of memory managed through a memory resource pool conceptually may be considered as unlimited, such that no admission control is performed at the time virtual memory is initially allocated (usually at virtual system activation time). Of course, the host system may experience memory shortages in later stages, such that ultimately the host system is no longer able to support the sum of virtual memory of all hosted virtual systems.

9.2 Implementation

This subclause details the requirements related to classes and their properties for implementations of the Memory Resource Virtualization Profile. The CIM Schema descriptions for any referenced element and its sub-elements apply.

The list of all methods covered by the Memory Resource Virtualization Profile is provided in 9.3. The list of all classes and their elements that are covered by this profile is provided in 9.5.

In references to CIM Schema properties that enumerate values, the numeric value is normative and the descriptive text following it in parentheses is informative. For example, in the statement “If an instance of the CIM_ResourcePoolConfigurationCapabilities class contains the value 4 (DeleteResourcePool is supported) in an element of the SynchronousMethodsSupported[ ] array property”, the value “4” is normative text and “(DeleteResourcePool is supported)” is informative text.

9.2.1 Allocation units

All properties that describe storage extents of memory shall be measured in the allocation unit “kilobyte”. This requirement applies to all the following classes and properties:

- CIM_Memory
  - StartingAddress property: Value shall be in units of kilobyte
  - EndingAddress property: Value shall be in units of kilobyte

NOTE The properties NumberOfBlocks and ConsumableBlocks in the CIM_StorageExtent class that is base class of the CIM_Memory class allows specifying the memory size in units of blocks, with the blocksize specified through the value of the BlockSize property.

- CIM_ResourceAllocationSettingData
  - AllocationUnits property: Value shall be “byte*2^10” (equals kilobyte)
  - Reservation property: Value shall be in units of kilobyte
  - Limit property: Value shall be in units of kilobyte
- CIM_ResourcePool
  - AllocationUnits property: Value shall be “byte*2^10” (equals kilobyte)
  - Capacity property: Value shall be in units of kilobyte
  - Reserved property: Value shall be in units of kilobyte

9.2.2 Resource Allocation Profile

The Resource Allocation Profile described in clause 5 should be used to model host memory.

The Resource Allocation Profile described in clause 5 shall be used to model:

- memory resource pool
- memory resource allocation
- memory resource allocation request
- virtual memory

9.2.2.1 Host memory

The support of the representation of host memory is optional.

NOTE The support for the representation of host memory is mandatory if the *System Memory Profile* (DSP1026) is supported; see 9.2.4.

If the representation of host memory supported, all of the following rules apply:

1) Host memory shall be represented by one or more instances of the CIM_Memory class that are associated with the instance of the CIM_System class that represents the host system through an instance of the CIM_SystemDevice association.
2) If the host memory is composed from more than one extent, host memory shall be represented as a memory composition as follows:
   - Each composing memory extent shall be represented by an instance of the CIM_Memory class as required by rule 1).
   - Total memory shall be represented by an instance of the CIM_Memory class as required by rule 1). In that instance the value of the StartingAddress property shall be 0, and the value of the EndingAddress property shall be the highest ending address from any of the composing memory extents. The range spanned by subtracting the value of the EndingAddress property from that of the StartingAddress property shall be reflected by the values of the BlockSize, NumberOfBlocks and ConsumableBlocks properties, respectively.
   - Each instance if the CIM_Memory class representing a composing memory extent shall be associated with the instance of the CIM_Memory class representing total memory through an instance of the CIM_Component association.

NOTE In a memory composition, total memory may span memory gaps that are not covered by a composing memory extent.

9.2.2.2 Memory resource pool

This subclause specifies implementation requirements for the representation of memory resource pools.

9.2.2.2.1 CIM_ResourcePool.Primordial property

The value of the Primordial property shall be set to TRUE for any instance of the CIM_ResourcePool class that represents a primordial memory resource pool. For other instances of the CIM_ResourcePool class that represent memory resource pools, the value of the Primordial property shall be set to FALSE.

9.2.2.2.2 CIM_ResourcePool.PoolID property

The value of the PoolID property shall be set such that it enables unique identification of the instance of the CIM_ResourcePool class within the scoping host system.

9.2.2.2.3 Aggregation of host resources

The support of the representation of the aggregation of host memory into a primordial memory resource pool is optional.

If the representation of the aggregation of host memory into primordial memory resource pools is supported, at least one instance of the CIM_Component association (see 9.5.5) references the instance of the CIM_ResourcePool class that represents the pool.

9.2.2.2.4 CIM_ResourcePool.Reserved property

The value of the Reserved property shall denote the amount of host memory that is actually reserved from the resource pool, in units of kilobyte.

9.2.2.2.5 CIM_ResourcePool.Capacity property

The support of the Capacity property is conditional.

Conditional Requirement: The Capacity property shall be supported if the aggregation of host resources is supported (see 9.2.2.2.3); otherwise, support of the Capacity property is optional.

If the Capacity property is supported, its value shall reflect the maximum amount of memory that can be allocated from the resource pool in units of kilobyte. If the instance of the CIM_ResourcePool class represents a memory resource pool with unlimited capacity, the value of the Capacity property shall be set to the largest value supported by the uint64 datatype.

9.2.2.2.6 Memory resource pool hierarchies

The support of representing memory resource pool hierarchies is optional.

If the representation of memory resource pool hierarchies is supported, any concrete memory resource pool shall be represented through an instance of the CIM_ResourcePool class, where all of the following conditions shall be met:

- The value of the Primordial property shall be FALSE.
- The instance shall be associated through an instance of CIM_ElementAllocatedFromPool association to the instance of the CIM_ResourcePool class that represents its parent memory resource pool.
- The instance shall be associated through an instance of the CIM_ElementSettingData association to the instance of the CIM_ResourceAllocationSettingData class that represents the amount of memory allocated from the parent resource pool.

9.2.2.2.7 Default memory resource pool

The support of designating a default memory resource pool is optional.

If the designation of a default memory resource pool is supported, all of the following conditions apply:

- The default memory resource pool shall be represented by an instance of the CIM_ResourcePool class; see 9.5.18.
- That instance shall be associated to the instance of the CIM_AllocationCapabilities class that represents the pools default allocation capabilities as specified in 9.2.3.1.5.
- The same instance of the CIM_AllocationCapabilities class shall also represent the systems default allocation capabilities as specified in 9.2.3.1.2.

9.2.2.2.8 Memory resource pool management

The support of memory resource pool management is optional.

9.2.2.2.8.1 Indication of support

If memory resource pool management is supported, the instance of the CIM_ResourcePoolManagementCapabilities class that is associated through an instance of the CIM_ElementCapabilities association to the instance of the CIM_ResourcePoolManagementService that represents a memory resource pool configuration service shall represent the capabilities of the resource pool configuration service as specified in this subclause.

Memory resource pool management is supported (with one or more methods) if the SynchronousMethodsSupported[ ] array property, the AsynchronousMethodsSupported[ ] array property, or both have a non-NULL value and contain at least one element.

If memory resource pool management is not supported, the value of both the SynchronousMethodsSupported[ ] and the AsynchronousMethodsSupported[ ] array properties shall be NULL or an empty array in the instance of the CIM_ResourcePoolManagementCapabilities class that represents the capabilities of a resource pool configuration service.

9.2.2.2.8.2 ValueMap qualifier method designators

Qualifier values defined by the ValueMap qualifiers of the SynchronousMethodsSupported[ ] and AsynchronousMethodsSupported[ ] array properties in the CIM_ResourcePoolConfigurationCapabilities class shall designate methods as follows:

- The value 3 (CreateChildResourcePool is supported) designates the CreateChildResourcePool( ) method.
- The value 4 (DeleteResourcePool is supported) designates the DeleteResourcePool( ) method.
- The value 5 (AddResourcesToResourcePool is supported) designates the AddResourcesToResourcePool( ) method.
- The value 6 (RemoveResourcesFromResourcePool) designates the RemoveResourcesFromResourcePool( ) method.

9.2.2.2.8.3 Implementation requirements

Elements in the value sets of the SynchronousMethodsSupported[ ] and AsynchronousMethodsSupported[ ] array properties in the CIM_ResourcePoolConfigurationCapabilities class shall be specified according to the following rules:

- A particular ValueMap qualifier value shall appear as an element in the value set of at most one array property.

- If a particular ValueMap qualifier value does not appear in the value set of either array property, the corresponding method is not supported.
- If a particular qualifier value is used as element in the value set of the SynchronousMethodsSupported[ ] array property, the corresponding method shall be supported with synchronous behavior only.
- If a particular qualifier value is used as element in the value set of the AsynchronousMethodsSupported[ ] array property, the corresponding method shall be supported with synchronous behavior, asynchronous behavior, or both.

A method implementation shall apply following rules:

- A supported method (with synchronous or asynchronous behavior) shall not return 1 (Not Supported).
- A method supported with synchronous behavior only shall not return 4096 (Method Parameters Checked – Job Started).

9.2.2.2.8.4 Availability of support of asynchronous operations

This subclause specifies the CIM elements that indicate the availability of the support for asynchronous operations for memory resource pool management.

Asynchronous operations for memory resource pool management are supported (with one or more methods) if the AsynchronousMethodsSupported[ ] array property has a non-NULL value and contains at least one element.

9.2.2.3 Memory resource allocation

NOTE The Resource Allocation Profile described in clause 5 specifies two alternatives for modeling resource allocation: *simple resource allocation* and *virtual resource allocation*.

Implementations conforming to the Memory Resource Virtualization Profile shall implement virtual resource allocation as defined in 5.2.2.

9.2.2.4 CIM_ResourceAllocationSettingData

This subclause specifies the use of the CIM_ResourceAllocationSettingData class.

9.2.2.4.1 General

Instances of the CIM_ResourceAllocationSettingData class

- shall be used to represent memory resource allocation requests (MRQ)
- shall be used to represent memory resource allocations (MRA)
- shall be used to represent settings that define the capabilities of systems and of memory resource pools (MCA)
- may be used to represent settings that define the mutability of memory resource allocations (MMS)

The specifications in this subclause generally define constraints for property values used in these representations. Constraints that apply to only a subset of these representations are prefixed with the respective acronyms and followed by the word “Only” and a colon.

9.2.2.4.2 CIM_ResourceAllocationSettingData.PoolID property

The value of the PoolID property shall designate the memory resource pool. A NULL value shall indicate the use of the host system's default memory resource pool.

9.2.2.4.3 CIM_ResourceAllocationSettingData.ConsumerVisibility property

The value of the ConsumerVisibility property shall denote whether host memory is directly passed through to the virtual system or whether memory is virtualized. Values shall be assigned as follows:

- A value of 2 (Passed-Through) shall denote that host memory is passed through.
- A value of 3 (Virtualized) shall denote that memory is virtualized.
- MRQ Only: A value of 0 (Unknown) shall be used if the represented memory resource allocation request does not predefine which type of memory shall be allocated.

Other values shall not be used.

9.2.2.4.4 CIM_ResourceAllocationSettingData.HostResource[ ] Array property

Support of the HostResource[ ] array property is conditional if representing memory resource allocations, and optional otherwise.

MRA Only: Conditional Requirement: The HostResource[ ] array property shall be supported if the value of the MappingBehavior property is 2 (Dedicated).

Otherwise, the HostResource[ ] array property may be supported.

If HostResource[ ] array property is supported, the following rules apply:

- If the value of the ConsumerVisibility property (see 9.2.2.4.3) is 2 (Passed-Through), the value of the HostResource[ ] array property should designate the host memory resource that is passed through to a virtual system. A value of NULL or an empty array should indicate that the passed-through host memory resource is not represented by an instance of the CIM_Memory class.
- If the value of the ConsumerVisibility property is 3 (Virtualized), the value of the HostResource[ ] array property shall be NULL or shall be an empty array.
- MRA Only: The value of the HostResource[ ] array property depends on the value of the MappingBehavior property as follows:
  - If the value of the MappingBehavior property is 2 (Dedicated), elements in the value of the HostResource[ ] array property shall reference instances of the CIM_Memory class that represent host memory extents that are exclusively dedicated to the virtual system.
  - If the value of the MappingBehavior property is 3 (Hard Affinity) or 4 (Soft Affinity), elements in the value of the HostResource[ ] array property shall reference instances of the CIM_Memory class that represent host memory extents that provide the allocation of the virtual memory.
- MRQ Only: The value of the HostResource[ ] array property depends on the value of the MappingBehavior property as follows:
  - If the value of the MappingBehavior property is 2 (Dedicated), elements in the value of the HostResource[ ] array property shall reference instances of the CIM_Memory class that represent host memory extents that are required with dedicated access by the virtual system.
  - If the value of the MappingBehavior property is 3 (Soft Affinity), elements in the value of the HostResource[ ] array property shall reference instances of the CIM_Memory class that

represent host memory extents that are preferred for the allocation of the virtual systems memory.

– If the value of the MappingBehavior property is 4 (Hard Affinity), elements in the value of the HostResource[ ] array property shall reference instances of the CIM_Memory class that represent host memory extents that are required for the allocation of the virtual systems memory.

If the HostResource[ ] array property is not supported, the value of the HostResource[ ] array property shall be NULL. This indicates that host memory extents that are exclusively dedicated to the virtual system or that provide the allocation of the virtual memory are not defined.

9.2.2.4.5 CIM_ResourceAllocationSettingData.VirtualQuantity property

The value of the VirtualQuantity property shall denote the amount of virtual memory available to a virtual system in units of kilobyte.

9.2.2.4.6 CIM_ResourceAllocationSettingData.Reservation property

Support of the Reservation property is optional.

If the Reservation property is supported, the value of the Reservation property shall denote the amount of host memory reserved for the exclusive use of a virtual system in units of kilobyte.

If the Reservation property is not supported, it shall have a value of NULL. This indicates that an amount of host memory reserved for the exclusive use of the virtual system is not defined.

9.2.2.4.7 CIM_ResourceAllocationSettingData.Limit property

Support of the Limit property is optional.

If the Limit property is supported, the value of the Limit property shall denote the maximum amount of host memory available to a virtual system in units of kilobyte.

If the Limit property is not supported, it shall have a value of NULL. This indicates that a maximum amount of host memory available to the virtual system is not defined.

9.2.2.4.8 CIM_ResourceAllocationSettingData.Weight property

Support of the Weight property is optional.

If the Weight property is supported, its value shall denote the relative priority of a memory resource allocation in relation to other memory resource allocations.

If the Weight property is not supported, it shall have a value of NULL. This indicates that a relative priority of the memory resource allocation in relations to other memory resource allocations is not defined.

9.2.2.4.9 CIM_ResourceAllocationSettingData.Parent property

The implementation of the Parent property is optional.

If the Parent property is supported, the value of the Parent property shall denote the parent entity of the memory resource allocation.

If the Parent property is not supported, is shall have a value of NULL. This indicates that a parent entity of the memory resource allocation is not defined.

NOTE For example, the value of the Parent property may refer to the name of an address space that resides in the host system.

9.2.2.4.10 CIM_ResourceAllocationSettingData.Connection[ ] Array property

The implementation of the Connection[ ] array property is optional.

If Connection[ ] array property is supported, its value shall contain elements that identify entities connected to the memory resource allocation.

If the Connection[ ] array property is not supported, it shall have a value of NULL. This indicates that entities connected to the memory resource allocation are not defined.

NOTE For example, elements of the value of the Connection[ ] array property may refer to the name of shared memory segments that are mapped to the allocated virtual memory.

9.2.2.4.11 CIM_ResourceAllocationSettingData.MappingBehavior property

The implementation of the MappingBehavior property is optional.

If the MappingBehavior property is supported, its value shall denote how host resources referenced by elements in the value of HostResource[ ] array property relate to the memory resource allocation. The following rules apply:

- MRA Only:
  - A value of 2 (Dedicated) shall indicate that the represented memory resource allocation is provided by host memory resources as referenced by the value of the HostResource[ ] array property that are exclusively dedicated to the virtual system.
  - A value of 3 (Soft Affinity) or 4 (Hard Affinity) shall indicate that the represented memory resource allocation is provided using host memory resource as referenced by the value of the HostResource[ ] array property.
  - Other values shall not be used.
- MRQ Only:
  - A value of 0 (Unknown) shall indicate that the memory resource allocation request does not require specific host resources.
  - A value of 2 (Dedicated) shall indicate that the memory resource allocation request shall be provided by exclusively dedicated host memory resources as specified through the value of the HostResource[ ] array property.
  - A value of 3 (Soft Affinity) shall indicate that the memory resource allocation request shall preferably be provided by host memory resources as specified through the value of the HostResource[ ] array property, but that other resources may be used if the requested resources are not available.
  - A value of 4 (Hard Affinity) shall indicate that the memory resource allocation request shall preferably be provided by host memory resources as specified through the value of the HostResource[ ] array property and that other resources shall not be used if the requested resources are not available.
  - Other values shall not be used.

If the MappingBehavior property is not supported, it shall have a value of NULL. This indicates that a further qualification of the value of the HostResource[ ] array property through the value of the MappingBehavior property is not defined.

9.2.2.5 Virtual memory

For the representation of virtual memory all of the following rules apply:

1) Virtual memory shall be represented by one or more instances of the CIM_Memory class that are associated with the instance of the CIM_ComputerSystem class that represents the virtual system through an instance of the CIM_SystemDevice association.

2) If the virtual memory is composed from more than one extent, virtual memory shall be represented as a memory composition as follows:

   – Each composing memory extent shall be represented by an instance of the CIM_Memory class as required by rule 1).

   – Total memory shall be represented by an instance of the CIM_Memory class as required by rule 1). In that instance the value of the StartingAddress property shall be 0, and the value of the EndingAddress property shall be the highest ending address from any of the composing memory extents. The range defined by subtracting the value of the EndingAddress property from that of the StartingAddress property shall be reflected by the values of the BlockSize, NumberOfBlocks and ConsumableBlocks properties, respectively.

   – Each instance of the CIM_Memory class representing a composing memory extent shall be associated with the instance of the CIM_Memory class representing total memory through an instance of the CIM_Component association.

3) If a memory extent is directly based on a memory resource allocation, the instance of the CIM_Memory class representing that extent shall be associated to all of the following instances:

   – the instance of the CIM_ResourceAllocationSettingData that represents memory resource allocation through an instance of the CIM_SettingsDefineState association

   – the instance of the CIM_ResourcePool that represents the memory resource pool providing the resource allocation through an instance of the CIM_ElementAllocatedFromPool association

NOTE 1 In a memory composition, total memory may span memory gaps that are not covered by a composing memory extent. Discontiguous memory as seen in the memory address space presented to the virtual system is not supported.

NOTE 2 In a memory composition, total memory is never the direct result of a memory allocation; instead, each composing memory extent is the direct result of a memory allocation.

Additional constraints apply if DSP1026 is implemented; see 9.2.4.

9.2.3 Allocation Capabilities Profile

The Allocation Capabilities Profile described in clause 7 shall be used to model the following aspects:

- the memory resource allocation capabilities of host systems
- the memory resource allocation capabilities of memory resource pools
- the mutability of memory resource allocations or memory resource allocation requests

9.2.3.1 Memory resource allocation capabilities

The memory resource allocation capabilities of host systems and of memory resource pools shall be represented by instances of the CIM_AllocationCapabilities class.

9.2.3.1.1 CIM_AllocationCapabilities class (capabilities)

This subclause specifies the use of the CIM_AllocationCapabilities class for the representation of the memory resource allocation capabilities of host systems and of memory resource pools.

9.2.3.1.1.1 Relationship of system and resource pool capabilities

The memory allocation capabilities of a host system shall be a superset of the memory allocation capabilities of all memory resource pools that are hosted by the host system.

9.2.3.1.1.2 CIM_AllocationCapabilities.RequestedTypesSupported property

The value of the RequestedTypesSupported property shall indicate whether the host system or the resource pool support memory resource allocation requests for particular host resources, as follows:

- A value of 2 (Specific) shall indicate support of specific requests that only refer to a specific host memory resource.
- A value of 3 (General) shall indicate support of generic memory requests that do not refer to a particular host memory resource.
- A value of 4 (Both) shall indicate support of both specific and generic memory requests.

Other values shall not be used.

9.2.3.1.1.3 CIM_AllocationCapabilities.SharingMode property

The value of the SharingMode property shall indicate whether the host system or the resource pool support exclusive access or shared use of managed memory resources, as follows:

- A value of 2 (Dedicated) shall indicate support of memory resources with exclusive access. This value shall be used if host memory is allocated directly to a virtual system for exclusive use.
- A value of 3 (Shared) shall indicate support of memory resources with shared access. This value shall be used if host memory is not allocated directly to a virtual system.

Other values shall not be used.

NOTE More than one instance of the CIM_AllocationCapabilities class may be associated with a host system or resource pool. As a result, support of both shared and exclusive access to memory resources may be modeled, and one of these sharing modes may be designated the default sharing mode; see 9.2.3.1.3.

9.2.3.1.2 Host system memory resource allocation capabilities

Each instance of the CIM_System class that represents a host system shall be associated through the CIM_ElementCapabilities association with one or more instances of the CIM_AllocationCapabilities class that represent the host system's memory resource allocation capabilities. One of these instances shall represent the default memory resource allocation capabilities as specified in 9.2.3.1.3.

9.2.3.1.3 Default host system memory resource allocation capabilities

Exactly one instance of the CIM_AllocationCapabilities class shall exist that describes the default memory resource allocation capabilities of a host system. That instance shall be associated with the instance of the CIM_System class that represents the host system through an instance of the CIM_ElementCapabilities association where the value of the Characteristics[ ] array property shall contain exactly one element, and that element shall have a value of 2 (Default).

9.2.3.1.4 Memory resource pool memory resource allocation capabilities

Each instance of the CIM_ResourcePool class that represents a memory resource pool shall be associated through the CIM_ElementCapabilities association with one or more instances of the

CIM_AllocationCapabilities class that represent the memory resource pool's memory resource allocation capabilities. One of these instances shall represent the default memory resource allocation capabilities as specified in 9.2.3.1.5.

9.2.3.1.5 Default memory resource pool memory resource allocation capabilities

Exactly one instance of the CIM_AllocationCapabilities class shall exist that describes the default memory resource allocation capabilities of a memory resource pool. That instance shall be associated with the instance of the CIM_ResourcePool class that represents the memory resource pool through an instance of the CIM_ElementCapabilities where the value of the Characteristics array property shall contain exactly one element, and that element shall have a value of 2 (Default).

9.2.3.2 Memory resource allocation mutability

The support for the representation of the mutability of memory resource allocation requests and memory resource allocations is optional.

9.2.3.2.1 Indication of support

If the representation of the mutability of memory resource allocation requests and memory resource allocations is supported, in both cases it shall be represented by instances of the CIM_AllocationCapabilities class that are associated with the instance of the CIM_ResourceAllocationSettingData class that represents the memory resource allocation request or memory resource allocation through instances of the CIM_ElementCapabilities association.

If the representation of the mutability of a memory resource allocation or a memory resource allocation request is not supported, the instance of the CIM_ResourceAllocationSettingData class that represents the memory resource allocation request or the memory resource allocation shall not be associated with an instance of the CIM_AllocationCapabilities class through an instance of the CIM_ElementCapabilities association.

9.2.3.2.2 CIM_AllocationCapabilities class (mutability)

This subclause specifies the use of the CIM_AllocationCapabilities class for the representation of the mutability of a memory resource allocation request or a memory resource allocation.

9.2.3.2.2.1 CIM_AllocationCapabilities.RequestedTypesSupported property

The value of the RequestedTypesSupported property shall indicate whether the type of the memory resource allocation can be changed, as follows:

- A value of 2 (Specific) shall indicate support of change requests that refer to a specific host memory resource.
- A value of 3 (General) shall indicate support of change requests that do not refer to a particular host memory resource.
- A value of 4 (Both) shall indicate support of either type of change request.

Other values shall not be used.

9.2.3.2.2.2 CIM_AllocationCapabilities.SharingMode property

The value of the SharingMode property shall indicate whether the sharing mode of the memory resource allocation can be changed, as follows:

- A value of NULL shall indicate that the sharing mode of the memory allocation can not be changed.
- A value of 2 (Dedicated) shall indicate support of requests to change the sharing mode to exclusive access.
- A value of 3 (Shared) shall indicate support of requests to change the sharing mode to shared access.

Other values shall not be used.

9.2.3.2.2.3 CIM_AllocationCapabilities.SupportedAddStates[ ] Array property

If the addition of memory to the described memory resource allocation or memory resource allocation request is not supported, then If the value of the SupportedAddStates[ ] array property shall be NULL.

If the value set of the SupportedAddStates[ ] is empty, then the addition of memory to the described memory resource allocation or memory resource allocation request shall be supported regardless of the virtual system state of the scoping virtual system.

If values are provided as elements of the SupportedAddStates[ ] array property, these values shall designate a set of potential virtual system states. The addition of memory to the described memory resource allocation or memory resource allocation request shall be supported if the scoping virtual system is in any of the designated virtual system states.

9.2.3.2.2.4 CIM_AllocationCapabilities.SupportedRemoveStates[ ] Array property

If the removal of memory from the described memory resource allocation or memory resource allocation request is not supported, then the value of the SupportedRemoveStates[ ] array property shall be NULL.

If the value set of the SupportedRemoveStates[ ] array property is empty, then the removal of memory from the described memory resource allocation or memory resource allocation request shall be supported regardless of the virtual system state of the scoping virtual system.

If values are provided as elements of the SupportedRemoveStates[ ] array property, these values shall designate a set of potential virtual system states. The removal of memory from the described memory resource allocation or memory resource allocation request shall be supported if the scoping virtual system is in any of the designated virtual system states.

9.2.4 System memory profile

The support of DSP1026 for representation of host system memory is optional.

The support of DSP1026 for representation of virtual system memory is optional. Additional constraints defined in 9.2.2.5 apply.

NOTE DSP1026 defines that there is exactly one instance of the CIM_Memory class associated to the instance of the CIM_System class representing a system through an instance of the CIM_SystemDevice association. In context of the Memory Resource Virtualization Profile it is expected that there are host systems as well as virtual systems that have more than one memory extent assigned. A possible solution would be to use a memory composition where the instance of the CIM_Memory class that represents the memory composition is assigned as central instance of DSP1026.

9.3 Methods

This subclause details the requirements for supporting intrinsic operations and extrinsic methods for the CIM elements defined by the profile described in this clause.

9.3.1 General

Support of intrinsic and extrinsic methods is specified by the “Methods” clauses in 5.3 and 7.3. The only class added through the Memory Resource Virtualization Profile is the CIM_RegisteredProfile class; see 9.5.15.

This subclause details the requirements for supporting intrinsic operations and extrinsic methods for the CIM elements defined by the profile described in this clause.

9.3.2 Profile conventions for operations

For each profile class (including associations), the implementation requirements for operations, including for those in the following default list, are specified in class-specific subclauses of this subclause.

The default list of operations for all classes is:

- GetInstance( )
- EnumerateInstances( )
- EnumerateInstanceNames( )

For classes that are referenced by an association, the default list also includes

- Associators( )
- AssociatorNames( )
- References( )
- ReferenceNames( )

The implementation requirements for intrinsic operations and extrinsic methods of classes listed in 9.5, but not addressed by a separate subclause of this subclause, are specified by the "Methods" subclauses of respective base profiles, namely the Resource Allocation Profile described in clause 5 and the Allocation Capabilities Profile described in clause 7. These profiles are specialized by the Memory Resource Virtualization Profile, and in these cases this profile does not add method specifications beyond those defined in its base profiles.

9.3.3 CIM_RegisteredProfile

All operations in the default list in 9.3.2 are supported as described by DSP0200.

9.4 Use cases (informative)

The following use cases and object diagrams illustrate use of the profile described in this clause. They are for informative purposes only and do not introduce behavioral requirements for implementations of the profile.

9.4.1 Object diagram

Figure 31 depicts the CIM representation of a host system with one memory resource pool and one virtual system. Only information relevant in the context of memory resource virtualization is shown.

In Figure 31 the host system is represented by an instance of the CIM_System class tagged HOST. The host system owns two memory resources represented by instances of the CIM_Memory class that are tagged HOST_OWN and HOST_POOL. The first instance represents a memory extent in the lower part of the host memory that is used exclusively by the host because it is not assigned to a memory resource pool. The second instance represents a memory extent in the higher part of the host memory that is assigned to a memory resource pool.

The host system hosts a primordial memory resource pool represented by an instance of the CIM_ResourcePool class tagged MEM_POOL; in that instance the value of the PoolID property is “MEM_POOL”. The host memory resource that is represented by the instance of the CIM_Memory class tagged HOST_POOL is aggregated into the pool through an instance of the CIM_Component association.

The allocation capabilities of the host system and of the memory pool are represented by the same instance of the CIM_AllocationCapabilities class tagged CAP. Four instances of the CIM_ResourceAllocationSettingData class (tagged CAP_DEF, CAP_MIN, CAP_MAX, and CAP_INC) are associated with that instance through instances of the CIM_SettingsDefineCapabilities association. The values of the ValueRange and ValueRole properties in the association instances designate the referenced instances of the CIM_ResourceAllocationSettingData class that represent the default, minimum, maximum, and increment for memory resource allocations that are supported by the system and the pool.

The host system hosts a virtual system that is represented by an instance of the CIM_ComputerSystem class tagged VS. The hosted relationship is shown through an instance of the CIM_HostedDependency association.

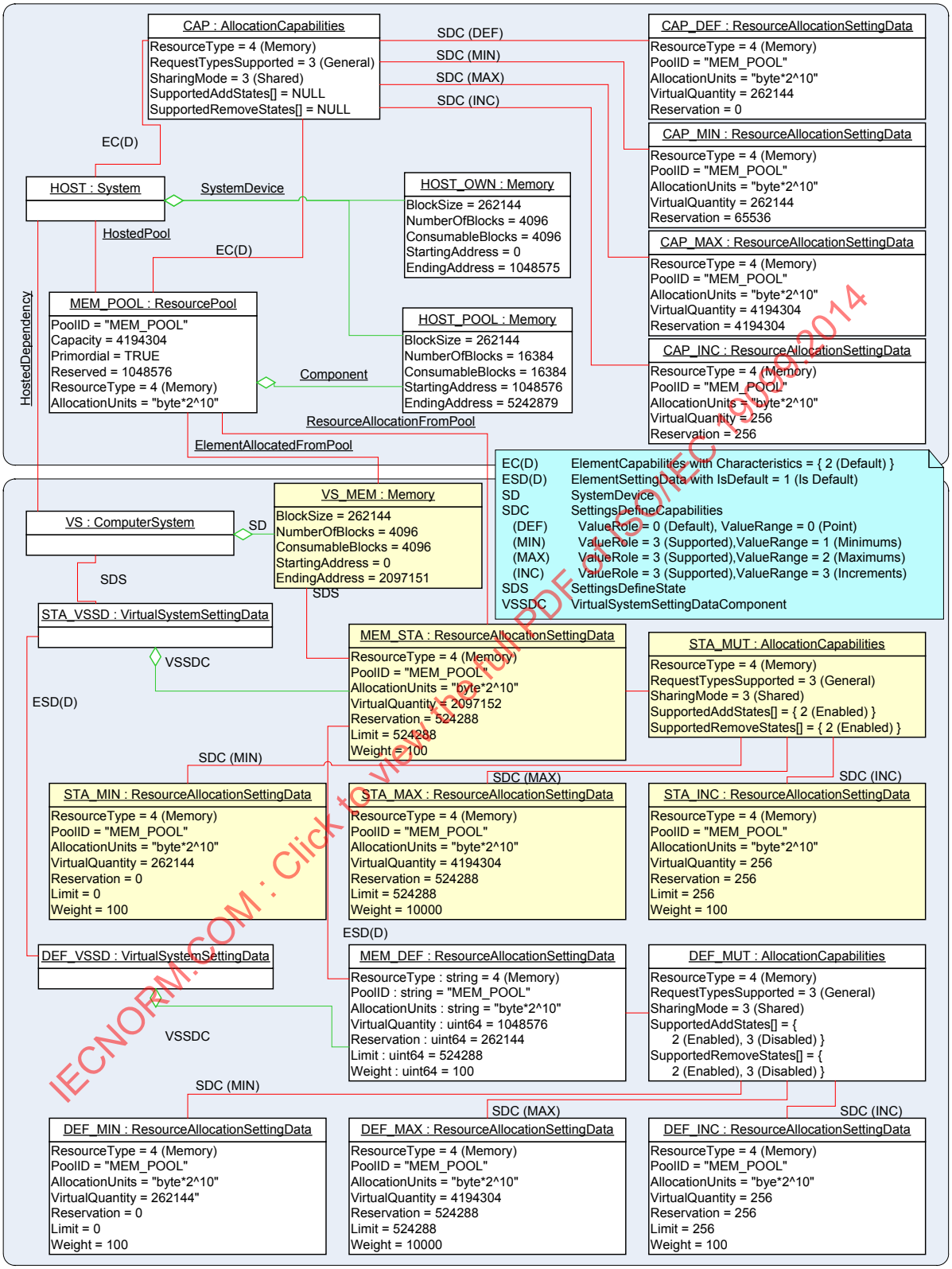

Figure 31 – Instance Diagram: Example CIM representation of memory resource virtualization

The head element of the “State” virtual system configuration is the instance of the CIM_VirtualSystemSettingData class tagged STA_VSSD; it is associated with the instance of the CIM_ComputerSystem class through an instance of the CIM_SettingsDefineState association. The “State” virtual system configuration contains an element of the CIM_ResourceAllocationSettingData class tagged MEM_STA that represents the memory resource allocation assigned to the virtual system. The virtual memory that results from the memory resource allocation is represented as part of the virtual system representation by the instance of the CIM_Memory class tagged VS_MEM.

NOTE All instances in Figure 31 that are marked with light yellow color represent “State” entities that exist only as long as the virtual system is active (that is, in a state other than “Defined”). These instances do not exist while the virtual system is in the “Defined” state (that is, it is not instantiated).

The head element of the “Defined” virtual system configuration is the instance of the CIM_VirtualSystemSettingData class tagged DEF_VSSD; it is associated with the head element of the “State” virtual system configuration through an instance of the CIM_ElementSettingData association where the value of the IsDefault property is 1 (Is Default). The “Defined” virtual system configuration contains an element of the CIM_ResourceAllocationSettingData class tagged MEM_DEF that represents the memory resource allocation request that defines the memory requirements of the virtual system. That definition is used when the virtual system is activated and host memory is allocated to support the virtual system’s virtual memory.

The example in Figure 31 shows a situation in which the VirtualQuantity property in the instances of the CIM_ResourceAllocationSettingData class in the “State” and the “Defined” virtual system configuration has different values. This indicates that the memory size was dynamically modified (for example, by an operator command) sometime after the virtual system activation. This reflects a temporary situation that is retained until the virtual system is recycled, at which point memory resources are deallocated and then newly allocated based on the memory resource allocation request in the virtual system definition.

The mutability of both the memory resource allocation request in the “Defined” virtual system configuration and of the memory resource allocation in the “State” virtual system configuration is represented by instances of the CIM_ResourceAllocationSettingData class associated through respectively parameterized instances of the CIM_SettingsDefineCapabilities association.

Acceptable virtual system states for the addition and the removal of virtual memory are different for the memory resource allocation request and the memory resource allocation. The memory resource allocation can be modified only while the virtual system remains instantiated, as indicated by a value of 2 (Enabled) in the instance of the CIM_AllocationCapabilities class tagged STA_MUT.

9.4.2 Inspection

This set of use cases describes how to obtain various CIM instances that represent memory-related information of host and virtual systems.

9.4.2.1 Obtain the memory size of an active virtual system

Assumption: All of the following:

- The client knows a reference to the instance of the CIM_ComputerSystem class that represents the virtual system.
- The virtual system is in a virtual system state other than “Defined” (that is, is instantiated), which is indicated by a value other than 3 (Disabled) for the EnabledState property in the instance of the CIM_ComputerSystem class.

The sequence of activities is as follows:

1) The client resolves the CIM_SystemDevice association to find instances of the CIM_Memory class that represent the virtual memory, invoking the intrinsic Associators( ) CIM operation with parameter values set as follows:

   – The value of the ObjectName parameter refers to the instance of the CIM_ComputerSystem class that represents the virtual system.

   – The value of the AssocClass parameter is set to “CIM_SystemDevice”.

   – The value of the ResultClass parameter is set to “CIM_Memory”.

   The result of step 1) is a set of instances of the CIM_Memory class.

2) For each of the instances returned from step 1), the client inspects the values of the BlockSize and NumberOfBlocks properties, multiplies these values, and adds the results. The resulting sum is the amount of virtual memory available to the virtual system.

Result: The client knows the amount of virtual memory available to the virtual system.

In the example CIM representation shown in Figure 31, the client initially would know the instance of the CIM_ComputerSystem class tagged VS that represents the virtual system. From there, the client would follow the CIM_SystemDevice association to the instance of the CIM_Memory class tagged VS_MEM that represents the only virtual memory resource allocated to the virtual system. The client would multiply the value of the BlockSize property (524288) with the value of the ConsumableBlocks property (4096), yielding a result of 2147483648 bytes or 2097152 KB for the virtual system memory size.

9.4.2.2 Obtain the memory size of a defined virtual system

Assumption: All of the following:

- The client knows a reference to the instance of the CIM_ComputerSystem class that represents the virtual system.
- The virtual system is in the “Defined” virtual system state (that is, is not instantiated), which is indicated by a value of 3 (Disabled) for the EnabledState property in the instance of the CIM_ComputerSystem class.

The sequence of activities is as follows:

1) The client resolves the CIM_SettingsDefineState association to find the instance of the CIM_VirtualSystemSettingData class that is the head element of the “State” virtual system configuration, invoking the intrinsic AssociatorNames( ) CIM operation with parameter values set as follows:

   – The value of the ObjectName parameter refers to the instance of the CIM_ComputerSystem class that represents the virtual system.

   – The value of the AssocClass parameter is set to “CIM_SettingsDefineState”.

   – The value of the ResultClass parameter is set to “CIM_VirtualSystemSettingData”.

   The result of step 4) is a reference to an instance of the CIM_VirtualSystemSettingData class.

2) The client obtains instances of the CIM_ElementSettingData association that reference the result instance from step 4) to find the instance of the CIM_VirtualSystemSettingData class that is the head element of the “Defined” virtual system configuration, invoking the intrinsic References( ) CIM operation with parameter values set as follows:

   – The value of the ObjectName parameter refers to the instance of the CIM_VirtualSystemSettingData class that is the head element of the “State” virtual system configuration.

– The value of the ResultClass parameter is set to “CIM_ElementSettingData”.

The result of this step is a set of instances of the CIM_ElementSettingData association. From that set, the client selects the only instance where the IsDefault property has a value of 1 (Is Default). The value of the SettingData property of that instance refers to the instance of the CIM_VirtualSystemSettingData class that is the head element of the “Defined” virtual system configuration.

3) The client resolves the CIM_VirtualSystemSettingDataComponent association to find instances of the CIM_ResourceAllocationSettingData class that represent resource allocation requests within the “Defined” virtual system configuration, invoking the intrinsic Associators( ) CIM operation with parameter values set as follows:

– The value of the ObjectName parameter refers to the instance of the CIM_VirtualSystemSettingData class that is the head element of the “Defined” virtual system configuration.

– The value of the AssocClass parameter is set to “CIM_VirtualSystemSettingDataComponent”.

– The value of the ResultClass parameter is set to “CIM_ResourceAllocationSettingData”.

The result of this step is a set of instances of the CIM_ResourceAllocationSettingData class that represent virtual resource allocation requests of the virtual system that are elements of the “Defined” virtual system configuration.

4) The client inspects the set of instances obtained in step 3), selecting only those instances where the ResourceType property has a value of 4 (Memory).

5) For each of the instances selected in step 4), the client then multiplies the of the VirtualQuantity property with 1024, yielding the amount of virtual memory requested through the inspected instance. The client builds the sum from the calculated values of all inspected instances with step 5).

NOTE Subclause 9.2.1 requires the value of the AllocationUnits property to be “byte*2^10”. Alternatively, the client might inspect the value of the AllocationUnits property in order to determine the unit.

Result: The client knows the amount of virtual memory requested for the virtual system.

In the example CIM representation shown in Figure 31, the client initially would know the instance of the CIM_ComputerSystem class tagged VS that represents the virtual system. From there, the client would follow the CIM_SettingsDefineState association to the instance of the CIM_VirtualSystemSettingData class tagged STA_VSSD. From there, the client would follow the CIM_ElementSettingData association where property IsDefault has a value of 1 (Is Default) to the instance of the CIM_VirtualSystemSettingData class tagged DEF_VSSD. Finally, from there the client would follow the CIM_VirtualSystemSettingDataComponent association to obtain the instance of the CIM_ResourceAllocationSettingData class tagged MEM_DEF that represents the only virtual memory resource allocation request for the virtual system.

The client would then multiply the value of the VirtualQuantity property with 1024, yielding a result of 1024*1048576 bytes or 1048576 KB requested for the virtual systems virtual memory.

9.4.2.3 Determine the allocation capabilities or allocation mutability

Assumption: Any of the following:

- The client knows a reference to an instance of the CIM_System class that represents a host system.
- The client knows a reference to an instance of the CIM_ResourcePool class that represents a memory resource pool.

- The client knows a reference to an instance of the CIM_ResourceAllocationSettingData class that represents a memory resource allocation request.
- The client knows a reference to an instance of the CIM_ResourceAllocationSettingData class that represents a memory resource allocation.

In the first two cases, this use case describes determining the related memory resource allocation capabilities; in the latter two cases, this use case describes determining the related mutability.

The sequence of activities is as follows:

1) The client resolves the CIM_ElementCapabilities association to find instances of the CIM_AllocationCapabilities class that represent allocation capabilities or mutability, invoking the intrinsic Associators( ) CIM operation with parameter values set as follows:

   – The value of the ObjectName parameter refers to the input instance.

   – The value of the AssocClass parameter is set to “CIM_ElementCapabilities”.

   – The value of the ResultClass parameter is set to “CIM_AllocationCapabilities”.

   The result of this step is a set of instances of the CIM_AllocationCapabilities class that represent allocation capabilities or mutability.

2) The client cycles through the set of instances of the CIM_AllocationCapabilities class obtained in step 1), fetching instances of the CIM_SettingsDefineCapabilities association that reference each of the instances from step 1) by invoking the intrinsic References() CIM operation with parameter values set as follows:

   – The value of the ObjectName parameter each cycle refers another instance of the CIM_AllocationCapabilities class from the result set of step 1).

   – The value of the ResultClass parameter is set to “CIM_SettingsDefineCapabilities”.

   The result of this step each time is a set of instances of the CIM_SettingsDefineCapabilities association.

3) For each of the instances obtained in step 2), the client inspects the values of the ValueRole and ValueRange properties to determine the type of limitation imposed by the instance of the CIM_ResourceAllocationSettingData class referenced by the value of the PartComponent property in that association instance, as follows:

   – A default setting is designated through a value of 0 (Default) for the ValueRole property and a value of 0 (Point) for the ValueRange property. A default setting does not apply for the description of mutability.

   – A minimum setting is designated through a value of 3 (Supported) for the ValueRole property and a value of 1 (Minimums) for the ValueRange property.

   – A maximum setting is designated through a value of 3 (Supported) for the ValueRole property and a value of 2 (Maximums) for the ValueRange property.

   – An increment setting is designated through a value of 3 (Supported) for the ValueRole property and a value of 3 (Increments) for the ValueRange property.

4) The client obtains for each of the instances obtained in step 2) the referenced instance of the CIM_ResourceAllocationSettingData class, invoking the intrinsic GetInstance( ) CIM operation with the value of the InstanceName parameter set to the value of the PartComponent property from the association instance. That value refers to the instance of the CIM_ResourceAllocationSettingData class that represents the capabilities setting.

   The result is the instance of the CIM_ResourceAllocationSettingData class with the values of all non-null numeric properties that describe the settings.

Result: The client knows the memory allocation capabilities of the system or the memory resource pool, or the mutability of a memory resource allocation request or a memory resource allocation.

In the example CIM representation shown in Figure 31, the capabilities of the system and the resource pool are represented by the instance of the CIM_AllocationCapabilities class tagged CAP that is referenced likewise through an instance of the CIM_ElementCapabilities association from the instance of the CIM_System class tagged HOST that represents the system and from the instance of the CIM_ResourcePool class tagged MEM_POOL that represents a memory resource pool. Four instances of the CIM_ResourceAllocationSettingData class tagged CAP_DEF, CAP_MIN, CAP_MAX, and CAP_INC describe the applicable default, minimum, maximum, and increment settings that describe the allocation capabilities. These instances are all associated through respectively parameterized instances of the CIM_SettingsDefineCapabilities association with the instance of the CIM_AllocationCapabilities class tagged CAP.

In the example CIM representation shown in Figure 31, the mutability of the memory resource allocation request is represented by the instance of the CIM_AllocationCapabilities class tagged DEF_MUT. The mutability of the memory resource allocation is represented by the instance of the CIM_AllocationCapabilities class tagged DEF_STA. Values of elements in the value sets of the SupportedAddStates[ ] and SupportedRemoveStates[ ] array properties indicate the virtual system state for which respective additions and removals or memory resource allocation requests or memory resource allocations are supported. For both instances of the CIM_AllocationCapabilities class, three instances of the CIM_ResourceAllocationSettingData class are associated through respectively parameterized instances of the CIM_SettingsDefineCapabilities association that describe minimum, maximum, and increment settings.

9.4.2.4 Determine the default memory allocation capabilities

Assumption: The client knows a reference to the instance of the CIM_System class that represents the host system.

The sequence of activities is as follows:

1) The client obtains instances of the CIM_ElementCapabilities association that reference the instance of the CIM_System class, invoking the intrinsic References( ) CIM operation with parameter values set as follows:

   – The value of the ObjectName parameter refers to the instance of the CIM_System class.

   – The value of the ResultClass parameter is set to “CIM_ElementCapabilities”.

   The result of this step is a set of instances of the CIM_ElementCapabilities association.

2) From the result set of step 1), the client drops those instances where the value set of the Characteristics[ ] array property does not contain an element with the value 2 (Default).

   The result of this step is a set of instances of the CIM_ElementCapabilities association that reference instances of the CIM_AllocationCapabilities class that represent the default allocation capabilities of the system for a number of resource types.

3) For each of the association instances obtained in step 2), the client obtains the instance of the CIM_AllocationCapabilities class that is referenced by the value of the Capabilities property in the respective association instance, invoking the intrinsic GetInstance( ) CIM operation with the value of the InstanceName parameter set to the value of the Capabilities property.

   The result of this step is a set of instances of the CIM_ResourceAllocationSettingData class that represent the system’s default allocation capabilities for a number of resource types.

4) From the result set of step 3), the client drops those instances where the value set of the ResourceType property is not 4 (Memory).

   The result of this step is one instance of the CIM_ResourceAllocationSettingData class that represents the system's default allocation capabilities for the memory resource type. The client continues as in use case 9.4.2.3 step 2) in order to determine the set of instances of the CIM_ResourceAllocationSettingData that represent the settings for the default memory resource allocation capabilities.

Result: The client knows the default memory allocation capabilities of the system.

In the example CIM representation shown in Figure 31, the default allocation capabilities for the memory resource type of the system are represented by the instance of the CIM_AllocationCapabilities class tagged CAP.

9.4.2.5 Determine the default memory resource pool

Assumption: The client knows a reference to the instance of the CIM_AllocationCapabilities class that represents the default memory resource allocation capabilities of the system; see 9.4.2.4.

The sequence of activities is as follows:

1) The client obtains instances of the CIM_ElementCapabilities association that reference the instance of the CIM_AllocationCapabilities class, invoking the intrinsic References( ) CIM operation with parameter values set as follows:

   – The value of the ObjectName parameter refers to the instance of the CIM_AllocationCapabilities class.

   – The value of the ResultClass parameter is set to "CIM_ElementCapabilities".

   The result of this step is a set of instances of the CIM_ElementCapabilities association.

2) From the result set of step 1), the client drops those instances where the value set of the Characteristics[ ] array property does not contain an element with the value 2 (Default).

   The result of this step is a set of two instances of the CIM_ElementCapabilities association. One association instance references the instance of the CIM_ResourcePool class that represent the default memory resource pool, and one instance references the instance of the CIM_System class that represents the host system.

3) The client selects the instance of the CIM_ElementCapabilities association from the result of step 2) that references the instance of the CIM_ResourcePool class by comparing the value of the ManagedElement property against the known reference to the CIM_System class that represents the host system and dropping that association instance. The client uses the remaining association instance from the result set of step 2) to obtain the instance of the CIM_ResourcePool class that is referenced by the value of the ManagedElement property in that association instance, invoking the intrinsic GetInstance( ) CIM operation with the value of the InstanceName parameter set to the value of the ManagedElement property.

   The result of this step is the instance of the CIM_ResourcePool class that represents the system's default memory resource pool.

Result: The client knows the default memory resource pool of the system.

In the example CIM representation shown in Figure 31, the default memory resource pool is represented by the instance of the CIM_ResourcePool class tagged MEM_POOL.

9.4.2.6 Obtain the Memory Pool with the Largest Unreserved Capacity

Assumption: The client knows a reference to the instance of the CIM_System class that represents the host system.

The sequence of activities is as follows:

1) The client resolves the CIM_HostedPool association to find instances of the CIM_ResourcePool class that represent resource pools hosted by the host system, invoking the intrinsic AssociatorNames( ) CIM operation with parameter values set as follows:
   - The value of the ObjectName parameter refers to the instance of the CIM_System class that represents the host.
   - The value of the AssocClass parameter is set to “CIM_HostedPool”.
   - The value of the ResultClass parameter is set to “CIM_ResourcePool”.

   The result of this step is a set of instances of the CIM_ResourcePool class that represent resource pools hosted by the host system.
2) The client selects from the result set of step 1) only those instances where the value of the ResourceType property is 4 (Memory).

   The result is a set of instances of the CIM_ResourcePool class that represent memory resource pools hosted by the host system.
3) The client inspects the value of the Capacity and the Reserved properties in all instances selected with step 2), and each time calculates the amount of unreserved memory capacity by subtracting the value of the Reserved property from the value of the Capacity property.
4) From all pools inspected in step 3), the client selects the one that has the largest free capacity.
5) The client checks the resource pool selected in step 4) for architectural limitations as expressed by the pool’s capabilities, applying use case 9.4.2.3.

Result: The client knows the memory resource pool with the largest unreserved capacity.

NOTE The largest extent of memory actually available may be significantly smaller than indicated by the result because fragmentation may subdivide the amount of memory available into several smaller extents.

In the example CIM representation shown in Figure 31, the client initially would know the instance of the CIM_System class tagged HOST that represents the host system. From there, the client would follow the CIM_HostedPool association to instances of the CIM_ResourcePool class. Typically the association resolution would yield more than one instance, including instances that represent resource pools of other resource types, such that the client is required to select only those instances where the value of the ResourceType property is 4 (Memory). In Figure 31, there is only the instance of the CIM_ResourcePool class, tagged MEM_POOL, so the selection process is not required. From that instance, the client takes the value of the Capacity property and subtracts the value of the Reserved property (4194304 – 1048576) KB, yielding 3145728 KB as the maximum amount of memory potentially available from the pool.

9.5 CIM elements

Table 121 lists CIM elements that are defined or specialized for the profile described in this clause. Each CIM element shall be implemented as described in Table 121. The CIM Schema descriptions for any referenced element and its sub-elements apply.

Subclauses 9.2 (“Implementation”) and 9.3 (“Methods”) may impose additional requirements on these elements.

Table 121 – CIM Elements: Memory Resource Virtualization Profile

Element	Requirement	Description
Classes		
CIM_AffectedJobElement	Conditional	See 9.5.1.
CIM_AllocationCapabilities (Capabilities)	Mandatory	See 9.5.2.
CIM_AllocationCapabilities (Mutability)	Optional	See 9.5.3.
CIM_Component (Memory Composition)	Optional	See 9.5.4.
CIM_Component (Resource Pool)	Optional	See 8.5.1.
CIM_ConcreteJob	Conditional	See 9.5.6.
CIM_ElementAllocatedFromPool	Mandatory	See 8.5.2.
CIM_ElementCapabilities (Capabilities)	Mandatory	See 9.5.8.
CIM_ElementCapabilities (Mutability)	Conditional	See 9.5.9.
CIM_ElementCapabilities (Resource Pool)	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_ElementSettingData (Memory Pool)	Mandatory	See 8.5.5.
CIM_ElementSettingData (Memory Resource)	Mandatory	See 9.5.11.
CIM_HostedDependency	Optional	See 8.5.6.
CIM_HostedResourcePool	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_HostedService	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_Memory (Host System)	Conditional	See 8.5.7.
CIM_Memory (Virtual System)	Mandatory	See 8.5.8.
CIM_RegisteredProfile	Mandatory	See 8.5.9.
CIM_ResourceAllocationFromPool	Optional	See 8.5.10.
CIM_ResourceAllocationSettingData	Mandatory	See 8.5.11.
CIM_ResourcePool	Mandatory	See 8.5.12.
CIM_ResourcePoolConfigurationCapabilities	Mandatory	See 9.5.19.
CIM_ResourcePoolConfigurationService	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_SettingsDefineState	Mandatory	See 8.5.13.
CIM_ServiceAffectsElement	Mandatory	See 9.5.21.
CIM_SystemDevice (Virtual Memory)	Mandatory	See 9.5.22.
CIM_SystemDevice (Host Memory)	Optional	See 8.5.14.
Indications		
None defined		

9.5.1 CIM_AffectedJobElement

The support of the CIM_AffectedJobElement class is conditional.

Conditional Requirement: The CIM_AffectedJobElement association shall be supported if asynchronous operations for resource pool management are supported; see 9.2.2.2.8.4.

If the CIM_AffectedJobElement association is supported, instances of the CIM_AffectedJobElement association shall associate an instance of the CIM_ConcreteJob class that represents an asynchronous memory resource pool management task and all of the following:

- the instance of the CIM_ResourcePool class that represents a memory resource pool that is affected by the asynchronous memory resource pool management task
- the instance of the CIM_Memory class that represents host memory that is affected by the asynchronous memory resource pool management task

Table 122 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 122 – Association: CIM_AffectedJobElement

Elements	Requirement	Notes
AffectedElement	Mandatory	**Key:** Value shall reference an instance of the CIM_ResourcePool class or the CIM_Memory class. **Cardinality: ***
AffectingElement	Mandatory	**Key:** Value shall reference the instance of the CIM_ConcreteJob class. **Cardinality:** *

9.5.2 CIM_AllocationCapabilities (capabilities)

See 9.2.3.1.1 for detailed implementation requirements for this class if it is used for the representation of memory resource allocation capabilities of systems or of memory resource pools.

Table 123 lists the requirements for elements of this class in this case. These requirements are in addition to those specified in the CIM Schema and in the Allocation Capabilities Profile described in clause 7.

Table 123 – Class: CIM_AllocationCapabilities (memory allocation capabilities)

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
ResourceType	Mandatory	Value shall be 4 (Memory).
OtherResourceType	Mandatory	Value shall be NULL.
RequestTypesSupported	Mandatory	See 9.2.3.1.1.2.
SharingMode	Mandatory	See 9.2.3.1.1.3.
SupportedAddStates[ ]	Mandatory	Value shall be NULL.
SupportedRemoveStates[ ]	Mandatory	Value shall be NULL.

9.5.3 CIM_AllocationCapabilities (mutability)

The support of the CIM_AllocationCapabilities class for the representation of the mutability of memory resource allocation requests and memory resource allocations is optional.

If the CIM_AllocationCapabilities class is supported for the representation of the mutability of memory resource allocation requests and memory resource allocations, see 9.2.3.2.2 for detailed implementation requirements.

Table 124 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Allocation Capabilities Profile described in clause 7.

Table 124 – Class: CIM_AllocationCapabilities (memory allocation mutability)

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
ResourceType	Mandatory	Value shall be 4 (Memory).
OtherResourceType	Mandatory	Value shall be NULL.
RequestTypesSupported	Mandatory	See 9.2.3.2.2.1.
SharingMode	Mandatory	See 9.2.3.2.2.2.
SupportedAddStates[ ]	Optional	See 9.2.3.2.2.3.
SupportedRemoveStates[ ]	Optional	See 9.2.3.2.2.4.

9.5.4 CIM_Component (memory composition)

The support of the CIM_Component association for the representation of memory compositions is optional.

If the CIM_Component association is supported for the representation of memory compositions that are composed of other memory extents, instances of an association that is based on the CIM_Component association shall associate an instance of the CIM_Memory class that represents the memory composition and any instance of the CIM_Memory class that represent composing memory extents. If an implementation does not implement a more specific association based on the CIM_Component association, the CIM_ConcreteComponent association should be implemented.

NOTE The CIM_Component association is abstract; therefore it cannot be directly implemented. On the other hand, clients may directly follow abstract associations.

Table 125 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema.

Table 125 – Association: CIM_Component (memory resource)

Elements	Requirement	Notes
GroupComponent	Mandatory	**Key:** Value shall reference the instance of the CIM_Memory class that represents the memory composition. **Cardinality:** 0..1
PartComponent	Mandatory	**Key:** Value shall reference an instance of the CIM_Memory class that represents a composing memory extent. **Cardinality:** *

9.5.5 CIM_Component (resource pool)

The support of the CIM_Component association for the representation of memory extents that are aggregated by resource pools is optional.

If the CIM_Component association is supported for the representation of memory extents that are aggregated by resource pools, instances of an association that is based on the CIM_Component association shall associate an instance of the CIM_ResourcePool class that represents a primordial memory resource pool and any instance of the CIM_Memory class that represents host memory that is aggregated into the pool. If an implementation does not implement a more specific association based on the CIM_Component association, the CIM_ConcreteComponent association should be implemented.

NOTE The CIM_Component association is abstract; therefore it cannot be directly implemented. On the other hand, clients may directly follow abstract associations.

Table 126 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 126 – Association: CIM_Component (resource pool)

Elements	Requirement	Notes
GroupComponent	Mandatory	**Key:** Value shall reference the instance of the CIM_ResourcePool class that represents a memory resource pool. **Cardinality:** 0..1
PartComponent	Mandatory	**Key:** Value shall reference an instance of the CIM_Memory class that represents host memory aggregated into the pool. **Cardinality:** *

9.5.6 CIM_ConcreteJob

The support of the CIM_ConcreteJob class is conditional.

Conditional Requirement: The CIM_ConcreteJob class shall be supported if asynchronous operations for resource pool management are supported; see 9.2.2.2.8.4.

If the CIM_ConcreteJob is supported, instances of the CIM_ConcreteJob class shall represent asynchronous memory resource pool management tasks.

Table 127 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 127 – Class: CIM_ConcreteJob

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
DeleteOnCompletion	Mandatory	Value shall be TRUE.
ElementName	Mandatory	Value shall conform to pattern ".*".
ErrorCode	Mandatory	None
ErrorDescription	Mandatory	None
JobStatus	Mandatory	None
JobState	Mandatory	None

9.5.7 CIM_ElementAllocatedFromPool

An instance of the CIM_ElementAllocatedFromPool association shall associate an instance of the CIM_ResourcePool class that represents a memory resource pool with each instance of the CIM_Memory class that represents virtual memory resulting from a memory resource allocation out of the pool.

Table 128 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 128 – Association: CIM_ElementAllocatedFromPool

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Value shall reference the instance of the CIM_ResourcePool class that represents a memory resource pool. **Cardinality:** 1
Dependent	Mandatory	**Key:** Value shall reference the instance of the CIM_Memory class that represents virtual memory resulting from a memory allocation from the pool. **Cardinality:** *

9.5.8 CIM_ElementCapabilities (capabilities)

Instances of the CIM_ElementCapabilities association shall associate all of the following:

- the instance of the CIM_System class that represents the host system with each instance of the CIM_AllocationCapabilities class that represents memory allocation capabilities of the host system
- an instance of the CIM_ResourcePool that represents a memory resource pool with each instance of the CIM_AllocationCapabilities class that represents memory allocation capabilities of the memory resource pool

Table 129 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Allocation Capabilities Profile described in clause 7.

Table 129 – Association: CIM_ElementCapabilities (capabilities)

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Host: See 9.2.3.1.2 and 9.2.3.1.3. Pool: See 9.2.3.1.4 and 9.2.3.1.5. **Cardinality:** *
Capabilities	Mandatory	**Key:** Host: See 9.2.3.1.2 and 9.2.3.1.3. Pool: See 9.2.3.1.4 and 9.2.3.1.5. **Cardinality:** *
Characteristics	Mandatory	Host: See 9.2.3.1.2 and 9.2.3.1.3. Pool: See 9.2.3.1.4 and 9.2.3.1.5.

9.5.9 CIM_ElementCapabilities (mutability)

The support of the CIM_ElementCapabilities association for the representation of the mutability of a memory resource allocation request or a memory resource allocation is conditional.

Conditional Requirement: The support is required if the CIM_AllocationCapabilities class is supported for the representation of the mutability of memory resource allocation requests or memory resource allocations; see 9.5.3.

If the CIM_ElementCapabilities association is supported for the representation of the mutability of a memory resource allocation request or a memory resource allocation, an instance of the CIM_ElementCapabilities association shall associate an instance of the CIM_ResourceAllocationSettingData class that represents a memory resource allocation or memory resource allocation request with each instance of the CIM_AllocationCapabilities class that represents the mutability of the memory resource allocation or memory resource allocation request.

Table 130 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Allocation Capabilities Profile described in clause 7.

Table 130 – Association: CIM_ElementCapabilities (mutability)

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Value shall reference the instance of the CIM_ResourceAllocationSettingData class. **Cardinality:** *
Capabilities	Mandatory	**Key:** Value shall reference the instance of the CIM_AllocationCapabilities class. **Cardinality:** *
Characteristics[ ]	Mandatory	Value shall be { 3 (Current) }.

9.5.10 CIM_ElementSettingData (memory resource pool)

An instance of the CIM_ElementSettingData association shall associate an instance of the CIM_ResourcePool class that represents a concrete memory resource pool and the instance of the

CIM_ResourceAllocationSettingData class that represents the memory resource allocation that describes the allocation of the concrete resource pool from another resource pool.

Table 131 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 131 – Association: CIM_ElementSettingData (memory resource pool)

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Value shall reference the instance of the CIM_ResourcePool class that represents the concrete memory resource pool. **Cardinality:** 0..1
SettingData	Mandatory	**Key:** Value shall reference the instance of the CIM_ResourceAllocationSettingData class that represents the memory resource allocation. **Cardinality:** 0..1

9.5.11 CIM_ElementSettingData (memory resource)

An instance of the CIM_ElementSettingData association shall associate an instance of the CIM_ResourceAllocationSettingData class that represents a memory resource allocation and the instance of the CIM_ResourceAllocationSettingData class that represents the corresponding memory resource allocation request.

Table 132 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 132 – Association: CIM_ElementSettingData (memory resource)

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Value shall reference the instance of the CIM_ResourceAllocationSettingData class that represents the memory resource allocation. **Cardinality:** 1
SettingData	Mandatory	**Key:** Value shall reference the instance of the CIM_ResourceAllocationSettingData class that represents the memory resource allocation request. **Cardinality:** 0..1
IsDefault	Mandatory	Value shall be 1 (Is Default).
IsCurrent	Mandatory	Unspecified.
IsNext	Mandatory	Unspecified.

9.5.12 CIM_HostedDependency

The support of the CIM_HostedDependency association is optional.

If the CIM_HostedDependency association is supported, an instance of the CIM_HostedDependency association shall associate an instance of the CIM_Memory class that represents virtual memory with the instance of the CIM_Memory class that represents host memory that is dedicated for the support of the virtual memory.

Table 133 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 133 – Association: CIM_HostedDependency

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Value shall reference the instance of the CIM_Memory class that represents host memory. **Cardinality:** 0..1
Dependent	Mandatory	**Key:** Value shall reference the instance of the CIM_Memory class that represents virtual memory. **Cardinality:** 0..1

9.5.13 CIM_Memory (host system)

The support of the CIM_Memory class for the representation of host memory is conditional.

Conditional Requirement: The support is required if the CIM_SystemDevice association is supported for the representation of host memory; see 9.5.23.

Table 134 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in DSP1026 if that is implemented.

Table 134 – Class: CIM_Memory (host system)

Elements	Requirement	Notes
SystemCreationClassName	Mandatory	**Key**
CreationClassName	Mandatory	**Key**
SystemName	Mandatory	**Key**
Name	Mandatory	**Key**
EnabledState	Mandatory	Unspecified
RequestedState	Mandatory	Unspecified
StartingAddress	Mandatory	Unspecified
EndingAddress	Mandatory	Unspecified

9.5.14 CIM_Memory (virtual system)

See 9.2.2.5 for detailed implementation requirements for this class if it is used for the representation of virtual memory or virtual memory composition.

Table 135 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in DSP1026 if that is implemented.

Table 135 – Class: CIM_Memory (virtual system)

Elements	Requirement	Notes
SystemCreationClassName	Mandatory	**Key**
CreationClassName	Mandatory	**Key**
SystemName	Mandatory	**Key**
Name	Mandatory	**Key**
EnabledState	Mandatory	Unspecified
RequestedState	Mandatory	Unspecified
StartingAddress	Mandatory	Unspecified
EndingAddress	Mandatory	Unspecified

9.5.15 CIM_RegisteredProfile

The use of the CIM_RegisteredProfile class is specified by DSP1033.

Table 136 lists the requirements for elements of this class. These requirements are in addition to those specified in DSP1033.

Table 136 – Class: CIM_RegisteredProfile

Elements	Requirement	Notes
RegisteredOrganization	Mandatory	Value shall be set to 2 (DMTF).
RegisteredName	Mandatory	Value shall be set to “Memory Resource Virtualization”.
RegisteredVersion	Mandatory	Value shall be set to the version of the profile described in this clause: “1.0.0”.

9.5.16 CIM_ResourceAllocationFromPool

The support of the CIM_ResourceAllocationFromPool association is optional.

If the CIM_ResourceAllocationFromPool association is supported, an instance of the CIM_ResourceAllocationFromPool association shall associate an instance of the CIM_ResourcePool class that represents a memory resource pool with each instance of the CIM_ResourceAllocationSettingData class that represents a memory resource allocation from the pool.

Table 137 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 137 – Association: CIM_ResourceAllocationFromPool

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Value shall reference the instance of the CIM_ResourcePool class that represents a memory resource pool. **Cardinality:** 0..1

Elements	Requirement	Notes
Dependent	Mandatory	**Key:** Value shall reference the instance of the CIM_ResourceAllocationSettingData class that represents a memory resource allocation from the pool. **Cardinality:** *

9.5.17 CIM_ResourceAllocationSettingData

See 9.2.3.2.2 for detailed implementation requirements for this class.

Table 138 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 138 – Class: CIM_ResourceAllocationSettingData

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**; see the Resource Allocation Profile described in clause 5.
ResourceType	Mandatory	Value shall be 4 (Memory).
OtherResourceType	Mandatory	Value shall be NULL.
ResourceSubType	Optional	See the Resource Allocation Profile described in clause 5.
PoolID	Mandatory	See 9.2.2.4.2.
ConsumerVisibility	Optional	See 9.2.2.4.3.
HostResource[ ]	Optional	See 9.2.2.4.4.
AllocationUnits	Mandatory	See 9.2.1.
VirtualQuantity	Mandatory	See 9.2.2.4.5.
Reservation	Optional	See 9.2.2.4.6.
Limit	Optional	See 9.2.2.4.7.
Weight	Optional	See 9.2.2.4.8.
AutomaticAllocation	Optional	See the Resource Allocation Profile described in clause 5.
AutomaticDeallocation	Optional	See the Resource Allocation Profile described in clause 5.
Parent	Optional	See 9.2.2.4.9.
Connection[ ]	Optional	See 9.2.2.4.10.
MappingBehavior	Optional	See 9.2.2.4.11.

9.5.18 CIM_ResourcePool

Instances of the CIM_ResourcePool class shall represent memory resource pools.

Table 139 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 139 – Class: CIM_ResourcePool

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
ElementName	Optional	See the Resource Allocation Profile described in clause 5.
PoolID	Mandatory	See 9.2.2.2.2.
Primordial	Mandatory	See 9.2.2.2.1.
Capacity	Conditional	See 9.2.2.2.5.
Reserved	Optional	See 9.2.2.2.4.
ResourceType	Mandatory	Value shall be 4 (Memory).
OtherResourceType	Mandatory	Value shall be NULL.
ResourceSubType	Optional	See the Resource Allocation Profile described in clause 5.
AllocationUnits	Mandatory	See 9.2.1.

9.5.19 CIM_ResourcePoolConfigurationCapabilities

An instance of the CIM_ResourcePoolConfigurationCapabilities class shall represent the capabilities of a memory resource pool configuration service.

Table 140 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 140 – Class: CIM_ResourcePoolConfigurationCapabilities

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
AsynchronousMethodsSupported[ ]	Mandatory	See 9.2.2.2.8.
SynchronousMethodsSupported[ ]	Mandatory	See 9.2.2.2.8.

9.5.20 CIM_SettingsDefineState

An instance of the CIM_SettingsDefineState association shall associate an instance of the CIM_Memory class that represents virtual memory and the instance of the CIM_ResourceAllocationSettingData class that represents the memory resource allocation that yields the virtual memory.

Table 141 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 141 – Association: CIM_SettingsDefineState

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Value shall reference an instance of the CIM_Memory class. **Cardinality:** 0..1

Elements	Requirement	Notes
SettingData	Mandatory	**Key:** Value shall reference the instance of the CIM_ResourceAllocationSettingData class. **Cardinality:** 0..1

9.5.21 CIM_ServiceAffectsElement

An instance of the CIM_ServiceAffectsElement association shall associate an instance of the CIM_ResourcePoolConfigurationService class that represents a memory resource pool configuration service and each instance of the CIM_ResourcePool class that represents a memory resource pool that is configurable through the service.

Table 142 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 142 – Association: CIM_ServiceAffectsElement

Elements	Requirement	Notes
AffectedElement	Mandatory	**Key:** Value shall reference an instance of the CIM_ResourcePool class. **Cardinality:** *
AffectingElement	Mandatory	**Key:** Value shall reference the instance of the CIM_ResourcePoolConfigurationService class. **Cardinality:** 1

9.5.22 CIM_SystemDevice (virtual memory)

An instance of the CIM_SystemDevice association shall associate the instance of the CIM_ComputerSystem class that represents a virtual system and each instance of the CIM_Memory class that represents virtual memory in scope of the virtual system.

Table 143 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 143 – Association: CIM_SystemDevice (virtual memory)

Elements	Requirement	Notes
GroupComponent	Mandatory	**Key:** Value shall reference an instance of the CIM_System class. **Cardinality:** 1
PartComponent	Mandatory	**Key:** Value shall reference the instance of the CIM_Memory class. **Cardinality:** *

9.5.23 CIM_SystemDevice (host memory)

Support of the CIM_SystemDevice association for the representation of host memory is optional; see 9.2.2.1.

NOTE Support is mandatory if DSP1026 is implemented for the host system.

If the CIM_SystemDevice association is supported for the representation of host memory, an instance of the CIM_SystemDevice association shall associate the instance of the CIM_System class that represents the scoping host system and each instance of the CIM_Memory class that represents host memory in scope of the scoping host system.

Table 144 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema, in the Resource Allocation Profile described in clause 5, and in DSP1026 if that is implemented.

Table 144 – Association: CIM_SystemDevice (host memory)

Elements	Requirement	Notes
GroupComponent	Mandatory	**Key:** Value shall reference an instance of the CIM_System class. **Cardinality:** 1
PartComponent	Mandatory	**Key:** Value shall reference the instance of the CIM_Memory class. **Cardinality:** *

10 Storage Resource Virtualization Profile

Profile Name: Storage Resource Virtualization

Version: 1.0.0

Organization: DMTF

CIM Schema Version: 2.21

Central Class: CIM_ResourcePool

Scoping Class: CIM_System

The Storage Resource Virtualization Profile is a component profile that defines the minimum object model needed to provide for the CIM representation and management of the virtualization of storage extents or of disk drives.

Table 145 lists profiles on which the Storage Resource Virtualization Profile depends.

Table 145 – Related profiles for the Storage Resource Virtualization Profile

Profile Name	Organization	Version	Relationship	Description
Resource Allocation	DMTF	1.1	Specializes	The abstract profile that describes the virtualization of resources See clause 5.
Allocation Capabilities	DMTF	1.0	Specializes	The abstract profile that describes capabilities for resource allocation and resource mutability See clause 7.
Profile Registration	DMTF	1.0	Mandatory	The profile that specifies registered profiles
Indications	DMTF	1.0	Optional	The profile that specifies indications
Block Services	SNIA	1.3	Optional	The SMI-S package that describes block services
Host Discovered Resources	SNIA	1.2	Optional	The SMI-S profile that describes host discovered resources
Generic Initiator Ports	SNIA	1.0	Optional	The SMI-S profile that describes generic initiator ports
Generic Target Ports	SNIA	1.0	Optional	The SMI-S profile that describes generic target ports

Table 146 lists conditional and optional features defined in the profile described in this clause.

Table 146 – Optional Features

Feature Name	Requirement Level	Granularity	Description
Resource aggregation	Conditional	Instance of CIM_ResourcePool	The feature that defines the representation of the aggregation of host resources into host resource pools. See 10.2.5.
Resource pool management	Optional	Instance of CIM_ResourcePool-ConfigurationService	The feature that defines the management of resource pools. See 10.2.7.

NOTE Some elements adapted by the profile described in this clause are defined with a requirement level "conditional", with the condition referring to the implementation of a particular feature. This in effect requires the implementation of the conditional element if the feature is implemented.

10.1 Description

This subclause contains informative text only.

This subclause introduces the management domain addressed by the profile described in this clause, and outlines the central modeling elements established for representation and control of elements in the management domain.

10.1.1 General

In computer virtualization systems, virtual computer systems are composed of component virtual resources.

The profile described in this clause specializes the resource virtualization pattern as defined in the Resource Allocation Profile (described in clause 5) and the allocation capabilities pattern as defined in the Allocation Capabilities Profile (described in clause 7) for the representation and management of the following types of resources:

- virtual storage resources, designated by one of the resource type values 31 (Logical Disk), 32 (Storage Volume) or 19 (Storage Extent)
- virtual disk drives, designates by the value 17 (Disk Drive)

The profile described in this clause references additional CIM elements and establishes constraints beyond those defined in the referenced profiles.

Storage resources represented and managed by means of the profile described in this clause appear to an operating system running in the virtual computer system as virtual disks. Virtual disks either emulate "physical" disks (such as for example SCSI disks), or appear as "logical" disks that have no physical equivalent (such as for example block devices).

10.1.2 Storage resource virtualization class schema

Figure 32 shows the class schema of the profile described in this clause. It outlines the elements that are referenced and in some cases further constrained by the profile described in this clause, as well as the dependency relationships between elements of the profile described in this clause and other profiles. For simplicity in diagrams, the prefix *CIM_* has been removed from class and association names. Inheritance relationships are shown only to the extent required in the context of the profile described in this clause.

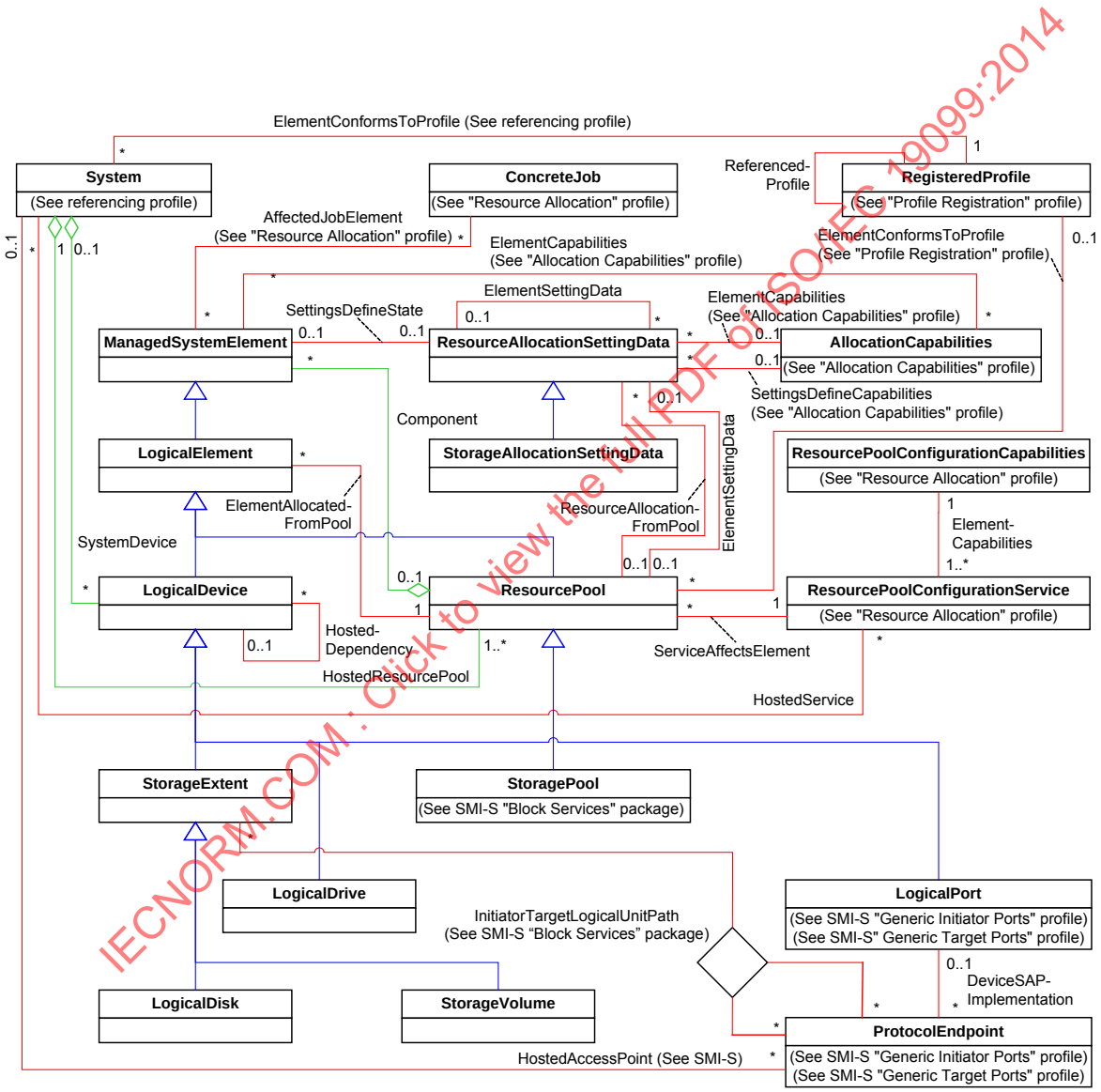

Figure 32 – Storage Resource Virtualization Profile: Profile class diagram

The profile described in this clause specifies the use of the following classes and associations:

- The CIM_ResourcePool class models resource pools for storage resources (such as storage volumes or logical disks) or for disk drives.
- The CIM_Component association models the relationship between resource pools and host storage resources as components of the resource pools.
- The CIM_ElementAllocatedFromPool association models hierarchies of resource pools and models the relationship of resource pools and storage resources allocated from those.
- The CIM_HostedResourcePool association models the hosting dependency between a resource pool and its host system. A host system supports at least one resource pool for storage resources.
- The CIM_LogicalDisk class, the CIM_StorageVolume class and the CIM_StorageExtent class model the following aspects of logical disks and storage volumes:
  - logical disks, storage volumes or plain storage extents as devices in the scope of a system, as modeled by the CIM_SystemDevice association
  - host storage extents (including subtypes) as components within storage resource pools, as modeled by the CIM_Component association
  - virtual disks as a result of a storage resource allocation from a storage resource pool, as modeled by the CIM_ElementAllocatedFromPool association
- The CIM_DiskDrive class models the following aspects of disk drives:
  - disk drives as devices in the scope of a system, as modeled by the CIM_SystemDevice association
  - disk drives as components within disk drive resource pools, as modeled through the CIM_Component association
  - disk drives as a result of a disk drive allocation from a disk drive resource pool, as modeled by the CIM_ElementAllocatedFromPool association
- The CIM_ResourceAllocationSettingData class models disk drive resource allocations or disk drive resource allocation requests
- The CIM_StorageAllocationSettingData class models storage resource allocations or storage resource allocation requests
- The CIM_AllocationCapabilities class and the CIM_ElementCapabilities association models
  - the resource allocation capabilities of host systems
  - the resource allocation capabilities of storage resource pools
  - the mutability of existing resource allocations
- The CIM_SettingsDefineCapabilities association models the relation between allocation capabilities and the settings that define these capabilities
- The CIM_ResourcePoolConfigurationService class models configuration services for resource pools and the CIM_ResourcePoolConfigurationCapabilities class modeling their capabilities
- The CIM_ConcreteJob class and the CIM_AffectedJobElement association models asynchronous management tasks initiated through resource pool configuration services

- The CIM_HostedDependency association models
  - the relationship between virtual storage extents and host storage extents
  - the relationship between virtual disk drives and host disk drives

10.1.3 Resource pools

This subclause describes the use of resource pools for storage resources and disk drives.

10.1.3.1 General

The profile described in this clause applies the concept of resource pools (defined in 5.1.1.2) to the following resource types:

- The resource type 31 (Logical Disk) designates storage resource pools that represent resources for the allocation of logical disks for immediate use by virtual systems; allocated logical disks are represented by instances of the CIM_LogicalDisk class
- The resource type 32 (Storage Volume) designates storage resource pools that represent resources for the allocation of storage volumes to virtual storage arrays; allocated storage volumes are represented by the instances of the CIM_StorageVolume class
- The resource type 19 (Storage Extent) designates storage resource pools that represent resources for the allocation of storage extents for virtual systems or virtual storage arrays that are not covered through resource types 31 (Logical Disk) or 32 (Storage Volume) as defined above
- The resource type 17 (Disk Drive) designates virtual disk drive storage pools that represent resources for the allocation of disk drives to virtual systems; allocated disk drives are represented by instances of the CIM_DiskDrive class

Note that the resource type of a resource pool governs the type of the resources that are allocated from the resource pool. Opposed to that the resource type of the resources that are aggregated by the resource pool may differ from the resource type of the pool. For example, a resource pool with a resource type of 31 (Logical Disk) supports the allocation of logical disks. However, the resources that are aggregated by that resource pool may be of a different type; for example, that resource pool might aggregate files, or it might represent a file system without representing individual files.

The profile described in this clause uses the resource pool as the focal point for storage resource allocations and disk drive allocations. Virtual systems receive storage resource allocations from storage resource pools based on storage resource allocation requests. Virtual systems receive disk drive resource allocations from disk drive resource pools based on disk drive resource allocation requests. In addition, a disk drive may also be allocated as a side effect of a storage resource allocation.

10.1.3.2 Representation of host resources

A resource pool represents host resources that enable the allocation of virtual devices (such as virtual disks or virtual disk drives). However the explicit representation of the host resources aggregated by a resource pool is optional: In some cases implementations may explicitly represent the host resources such as for example host logical disks, host storage volumes, host disk drives, files, file systems or file directories that are accessible by the host. In other cases implementations may choose not to explicitly represent the host resources aggregated by a resource pool. For example, an implementation that implements the representation and management of memory based virtual disks is not required to explicitly model the host memory that support the virtual disks. Instead, in this case the resource pool is the sole model element that represents host memory capacity assigned for the support of (allocated) virtual disks, and the host capacity that is still available for the allocation of new virtual disks.

The Resource Allocation Profile described in clause 5 defines two general types of resource pools: Primordial resource pools and concrete resource pools.

NOTE The SNIA SMIS:1.3, Part 3 *Block Devices*, *Block Services* package provides much stricter definitions of primordial storage pool and concrete storage pool than those of the Resource Allocation Profile described in clause 5. Implementations of the SNIA SMIS:1.3, Part 3 *Block Devices*, *Block Services* package for the management of host storage resources need to conform with these definitions. For example, the profile described in this clause allows the direct use of a primordial resource pool for the allocation of resources, while the SNIA SMIS:1.3, Part 3 *Block Devices*, *Block Services* package supports the creation and modification of storage volumes and logical disks only in context of concrete storage pools.

10.1.3.3 Primordial resource pool

A primordial resource pool aggregates capacity; it represents a subset of the manageable resources of a host system. Primordial resource pools are suitable to serve as the source of resource allocations — either for the allocation of child resource pools or for virtual resources.

10.1.3.4 Concrete resource pool

A concrete resource pool subdivides the capacity of its parent resource pool. The amount of capacity allocated to a concrete resource pool is less than or at most equal to the capacity of the parent pool.

10.1.3.5 Hierarchies of resource pools

The profile described in this clause specializes the concept of resource pool hierarchies defined in 5.1.1.4 to the resource types 31 (Logical Disk), 32 (Storage Volume), and 17 (Disk Drive).

Figure 33 shows an example of the CIM representation of a resource pool hierarchy where a set of host storage extents is aggregated into a primordial storage resource pool PRIM_POOL. The allocation capabilities of PRIM_POOL are represented by means of the Allocation Capabilities Profile described in clause 7. PRIM_POOL supports allocations starting from 4 GB up to 4 TB; in that range the increment is 1 KB.

Two host storage extents EXTENT1 and EXTENT2 are components of PRIM_POOL. The instances represent storage extents that are available to the host system. For example, the host system itself contains these storage devices such as for example local SCSI disks; or the host system has access to these devices through means of a storage area network or other network mechanisms.

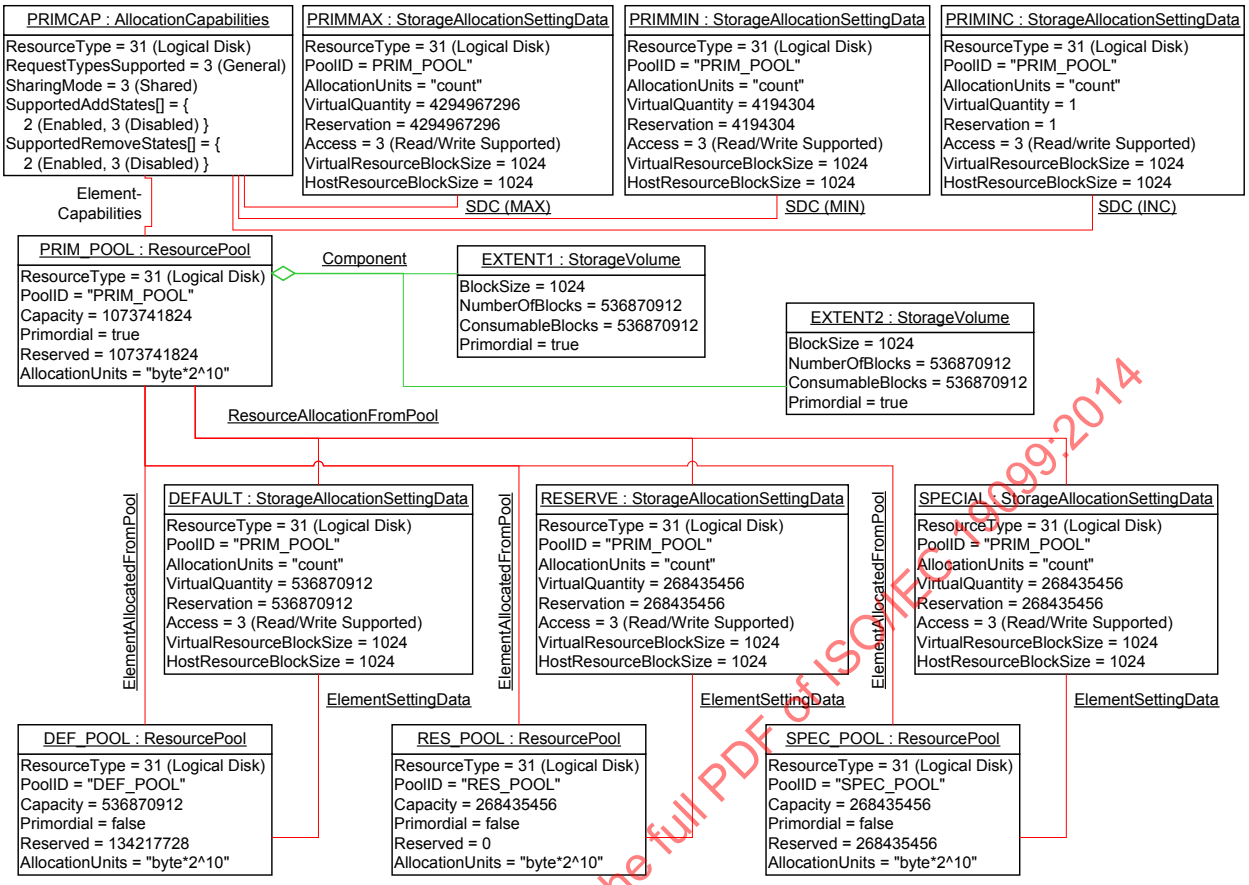

Figure 33 – Instance diagram: Concept of storage resource pool hierarchies

PRIM_POOL is subdivided into three concrete storage resource pools DEF_POOL, RES_POOL and SPEC_POOL, as follows:

- The concrete storage resource pool DEF_POOL represents a subextent of 512 GB. An amount of 128 GB is allocated out of that pool to support virtual disks that are not shown in Figure 33.
- The concrete storage resource pool RES_POOL represents a subextent of 256 GB. No allocations for virtual disks are drawn from this pool in the represented situation.
- The concrete storage resource pool SPEC_POOL represents a subextent of 256 GB. The complete capacity of SPEC_POOL is allocated to support virtual disks that are not shown in Figure 33.

10.1.3.6 Resource pool management

The profile described in this clause specializes the concept of resource pool management defined in 5.1.1.5 to the resource types 31 (Logical Disk), 32 (Storage Volume), 19 (Storage Extent), and 17 (Disk Drive).

10.1.4 Resource allocation

The profile described in this clause specializes the concept of device resource allocation defined in 5.1.1.3 to the resource types 31 (Logical Disk), 32 (Storage Volume), 19 (Storage Extent), and 17 (Disk Drive).

10.1.4.1 General

Depending on the resource type the result of a resource allocation as seen by a virtual system is either a virtual storage extent (including specializations such as a virtual disk or a virtual storage volume), or a virtual disk drive. In addition, the allocation of a virtual disk or a virtual storage volume may cause the allocation of a virtual disk drive as a side effect.

The representation of disk drives is optional. The profile described in this clause addresses three potential scenarios:

1) The allocation of a storage extent without explicit representation of a disk drive
2) The allocation of a storage extent with explicit representation of a disk drive

   An example is a virtual disk drive that is based on an image file stored in a host file. The disk drive may be implicitly allocated along with the allocation of the storage extent.
3) The allocation of a disk drive without modeling the allocation of a storage extent

   An example is a disk drive that is owned by a host system and is pathed through to a virtual system; in this case the media is volatile and cannot generally be modeled.

The profile described in this clause specifies the use of CIM_StorageAllocationSettingData class and the CIM_ResourceAllocationSettingData class such that all of these scenarios are covered.

For example, Figure 34 on page 228 shows a situation like in case 2) above. In this example the allocation of a logical disk to a virtual system causes the implicit allocation of a disk drive. Note that no separate RASD instance is required for allocation of the disk drive. Instead the implementation implicitly allocates the disk drive as part of the allocation of the logical disk and represents the disk drive by an instance of the CIM_DiskDrive class. The CIM_DiskDrive instance is associated to the instance of the CIM_LogicalDisk class representing the allocated logical disk by an instance of the CIM_MediaPresent association.

10.1.4.2 Storage resource allocations backed by files

In the example shown in Figure 34 the CIM_StorageAllocationSettingData instances directly refer to an image file through the value of the HostResource[ ] array property. In this example the value is formatted as a URI that encodes the file name. The value of the PoolID property refers to a resource pool that in this example represents a source for host files.

Implementations have various choices how to establish the relationship between host files and virtual disks, such as for example:

- Referring to preallocated files in the host environment
- Creation files as a side effect

An example of the latter case is depicted in Figure 34. In this case the implementation established a resource pool such for file-based logical disks. Allocations out of that resource pool are based on host files.

In this example the initial allocation of these host files is controlled by an implementation dependent rule that constructs the file location from several parts:

- a root path (such as for example "/var/vmfiles")
- a virtual system specific subpath (such as for example "vm1/disks" for a virtual system named "vm1"), and
- a disk specific file name (such as for example "imagedisk25.dsk").

The concatenation of the parts yields "/var/vmfiles/vm1/disks/imagedisk25.dsk". Note that such rules are highly implementation dependent. The model defined in the profile described in this clause facilitates the representation of rule-based assignment through the means of resource pools, but it does not specify elements that explicitly convey information about the rules themselves.

The profile described in this clause does not require implementations to expose information about resource availability or resource consumption in context of resource pools. For example, in Figure 34 in the CIM_ResourcePool instance FILEDISK_POOL the value of the Capacity property is NULL, indicating that the capacity of the resource pool is unknown to the implementation. Similarly, the value of the Reserved property is NULL, indicating that the amount of consumed resource is unknown to the implementation. It is expected that many implementations will be unaware of the amount of resource available through a resource pool because the resource pool is backed with resources from external units such as storage subsystems and network attached storage. In these situations the only purpose of the resource pool is to represent a particular source for resource allocations without requiring an implementation to have knowledge about resource capacity or consumption. A management client would have to contact the management interface of the external units in order to access respective information.

The example shown in Figure 34 also shows a representation of thin provisioning of a file-based logical disk. This is indicated by the value of the Limit property being defined and higher than the value of the Reservation property. In this example the value of the Reservation property requests an initial file size of 33554432 blocks (16 GB) up to a maximum file size of 134217728 blocks (64 GB) as expressed by the value of the Limit property; in both cases a block size of 512 applies as expressed by the value of the HostResourceBlockSize property. Opposed to that the disk size as seen by the virtual system (the consumer) remains constant at 64 GB; this is expressed by the value of the VirtualQuantity property, 16777216 blocks with a block size of 4096 as expressed by the value of the VirtualResourceBlockSize property. As the consuming virtual system starts writing data onto the virtual disk, for each logical 4-KB block of virtual disk a respective set of eight 512-KB blocks is allocated within the file. As soon as the amount of data to back the logical disk exceeds the initially requested file size (as expressed by the value of the Reservation property) the file starts growing beyond the initially assigned file size up to the size expressed by the value of the Limit property, such that finally when all blocks were at least once written by the virtual system the amount of storage provided equals the amount of storage consumed.

Note that in the example shown in Figure 34 the value of the Limit property expressed an amount of storage that is identical to that expressed by the value of the VirtualQuantity property. This implies that the file may grow until the complete virtual disk is backed with respective file data blocks. However, implementations may support placing restrictions on the file size; for example this may be the case in situations where a specific usage pattern such as a sparsely used disks is expected by the consumer. In this case if the upper file size as expressed by the value of the Limit property is reached, the consumer would receive a respective error indication.

Another concept applied in the example shown in Figure 34 is the remapping of blocks. In this example the block size at the providing host side is 512 (the value of the HostResourceBlockSize property), while the block size at the consuming virtual system side is 4096 (the value of the VirtualResourceBlockSize property).

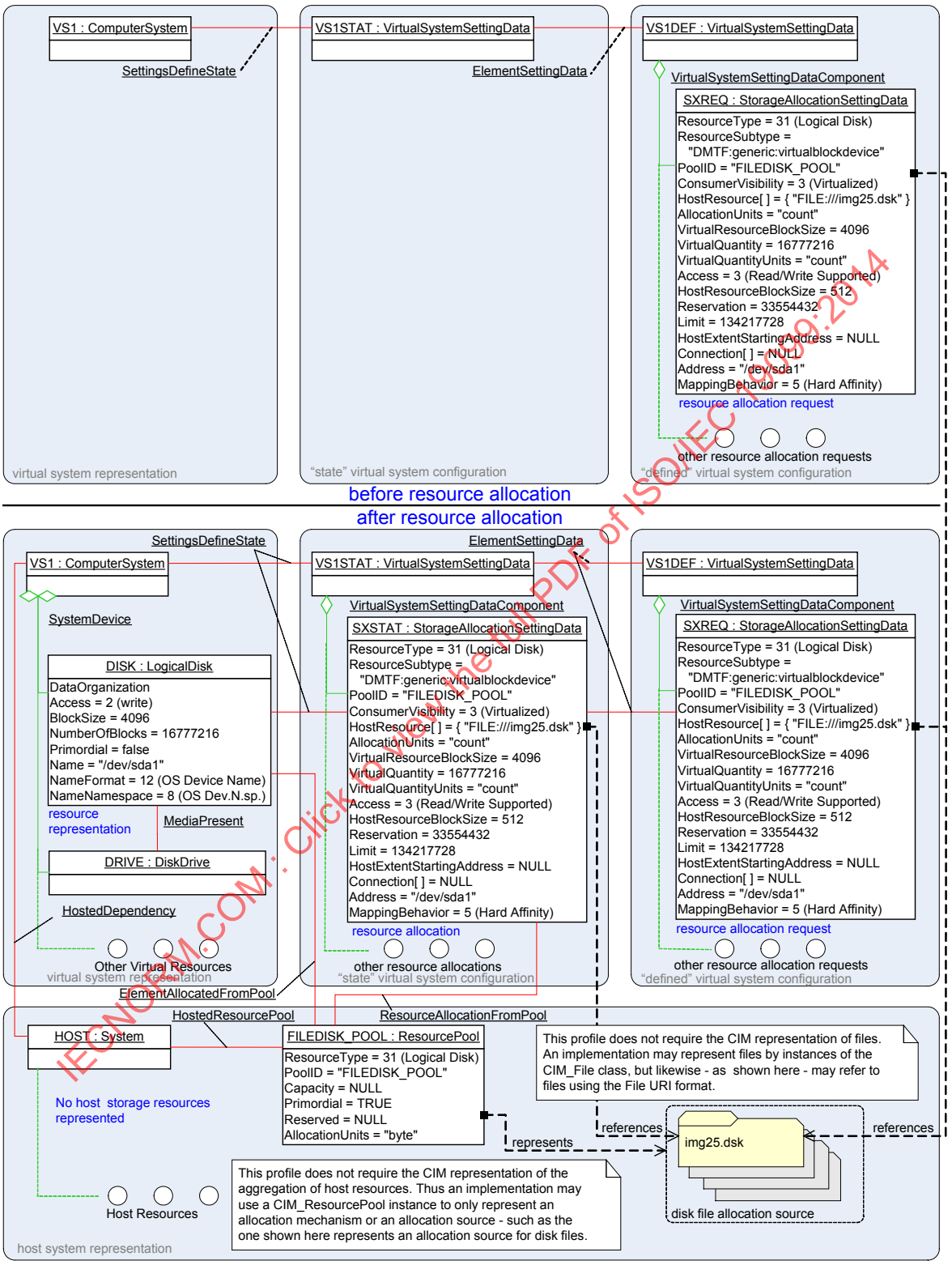

Figure 34 – Instance diagram: Concept of storage resource allocation

10.1.4.3 Resource allocation request

The resource requirements of a virtual system are represented by the "defined" virtual system configuration. (See the Virtual System Profile described in clause 12.) In a "defined" virtual system configuration disk drive resource allocation requests are represented by CIM_ResourceAllocationSettingData instances, and storage resource allocation requests are represented by CIM_StorageAllocationSettingData instances.

An example of the CIM representation of a storage resource allocation request is shown in the upper right part of Figure 34.

10.1.4.4 Resource allocation

As a virtual system is activated (instantiated), resources are allocated as requested by resource allocation requests in the "defined" virtual system definition. The actual resource allocations for a virtual system are represented by the "state" virtual system configuration. (See the Virtual System Profile described in clause 12.) In a "state" virtual system configuration disk drive resource allocations are represented by CIM_ResourceAllocationSettingData instances, and storage resource allocations are represented by CIM_StorageAllocationSettingData instances.

NOTE Storage resource allocation requests and storage resource allocations may directly reference persistent host resources — such as for example host storage extents or host files — through the value of the HostResource[ ] array property. These host resources persistently exist independent of their use as the base for virtual disks. However, there are situations where such host resources are unavailable at resource allocation time. For example, the file system that contains the file referenced by a storage resource allocation request might not be mounted, or a file might be in use by another consumer such as the host system itself or another virtual system. In these situations the resource allocation would fail at resource allocation time.

An example of the CIM representation of a storage resource allocation is shown in the center part of Figure 34.

10.1.4.5 Virtual disk

A virtual disk is the instantiation of resources allocated from a storage resource pool that is exposed to a virtual system through a logical device; it is the result of a storage resource allocation based on a storage resource allocation request.

Virtual disks may be virtualized or may be passed-through host storage resources.

An example of the CIM representation of a virtualized virtual disk as the result of a storage resource allocation is shown on the left side in the central part of Figure 34.

10.1.4.6 Virtual disk drive

A virtual disk drive is the instantiation of resources allocated from a resource pool that is exposed to a virtual system through a logical disk drive device; it is either the explicit result of a disk drive resource allocation based on a disk drive resource allocation request, or it is the implicit result of a storage resource allocation based on a storage resource allocation request.

Virtual disk drives may be virtualized or may be passed-through host disk drives. A virtual disk drive is represented by an instance of the CIM_DiskDrive class as part of the virtual system representation.

An example of the CIM representation of a virtual disk drive as the implicit result of a storage resource allocation is shown on the left side in the central part of Figure 34. In this case the virtual disk drive is implicitly allocated as a side effect of a storage resource allocation.

10.1.4.7 Storage virtualization

In the scope of the profile described in this clause, virtualization of storage is modeled through subdivision: Non overlapping sub-extents of a larger host storage extent may be assigned to different virtual systems. This allows subdividing a host storage extent for the use of a number of virtual systems.

10.1.4.8 Dedicated host storage

A dedicated storage extent is a storage extent owned or accessible by the host system that is exclusively reserved for support of a particular virtual disk of a particular virtual system.

10.1.4.9 Virtual storage extended configurability

Some types of virtual storage support extended configuration capabilities. For example virtual SCSI disks emulating physical SCSI disks may be grouped on virtual SCSI buses such that access is channeled from a virtual SCSI initiator port to a virtual SCSI target port, and from there to a target virtual LUN. An implementation may opt to represent ports and LUNs as modeled by respective profiles from SNIA SMI-S *Storage Management Technical Specification*, such as the Generic Initiator Ports profile or the Generic Target Ports profile described in SNIA SMIS Part 2 *Common Profiles* book, or more specific profiles that are based on these.

10.1.4.10 Management of host storage resources through SMI-S profiles

Implementation of the profile described in this clause may optionally implement profiles defined in the SNIA SMI-S *Storage Management Technical Specification*, such as the SNIA SMIS: Part 3 *Block Devices, Block Services* package or the *Extent Composition* subprofile for the management of host storage resources. An example of such a situation is depicted in Figure 35.

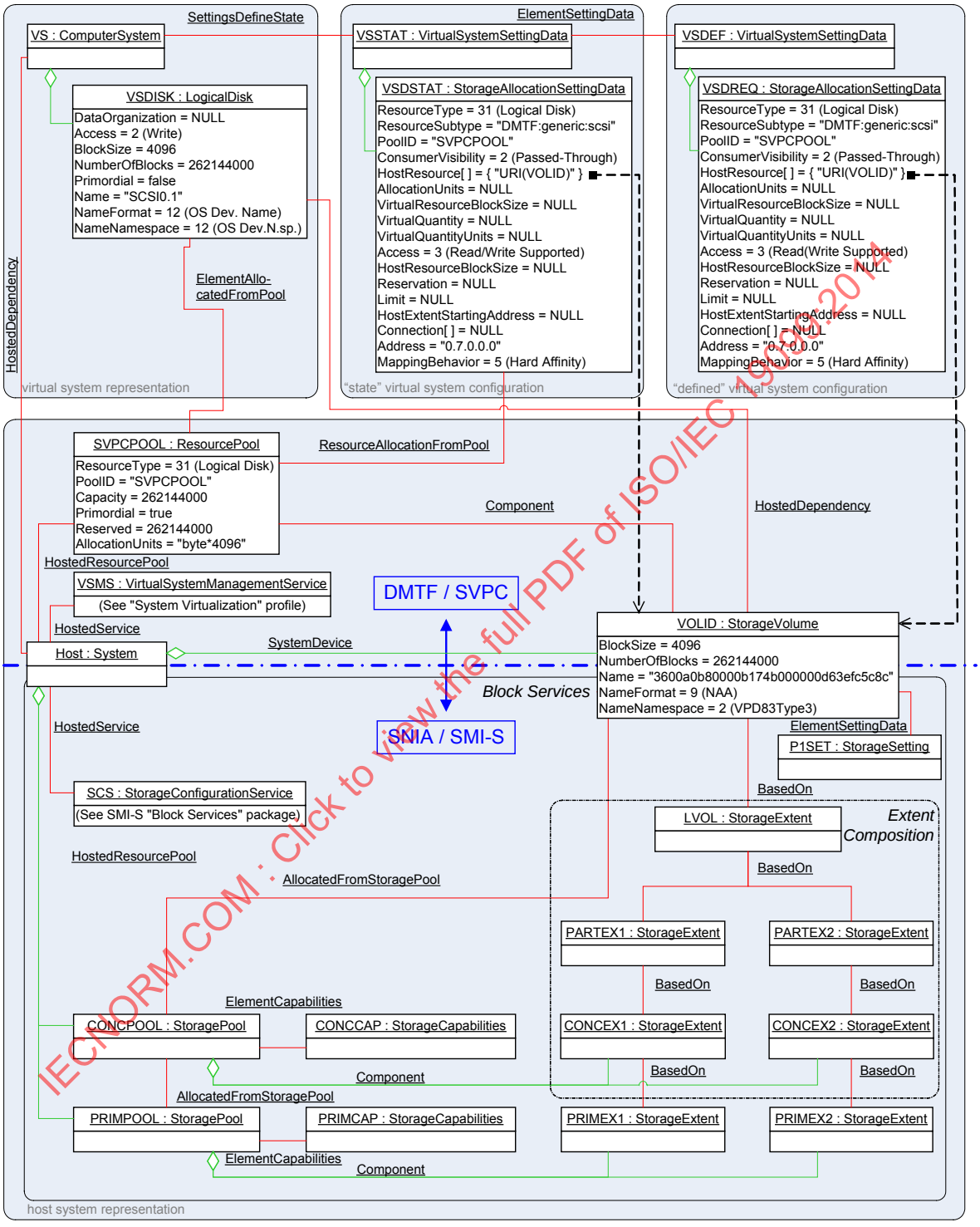

Figure 35 – Cooperation of DMTF SVPC and SNIA SMI-S profiles

The profile described in this clause (along with other system virtualization related profiles, such as the System Virtualization Profile described in clause 6) governs the allocation of storage resources to virtual systems. SMI-S profiles govern the allocation of host storage resources. The boundary between the model defined in the profile described in this clause and the models defined in SNIA SMIS is defined by CIM_StorageVolume instances such as VOLID in Figure 35. The configuration of these instances into the host environment may be represented and managed by means of SNIA SMIS, the configuration of virtual disks based on passed-through host disks is represented and managed by means of the profile described in this clause in coordination with other profiles of the SVPC suite of profiles. However, the use of SMI-S is optional in the context of the profile described in this clause; respective host resources may just as well be just discovered within the hosting environment by the implementation of the profile described in this clause.

The upper part of Figure 35 shows the configuration of an instantiated virtual system represented by the CIM_ComputerSystem instance VS1, with a logical disk represented by the CIM_LogicalDisk instance VSDISK. The logical disk is based on a passed-through host disk that is represented by the CIM_StorageVolume instance VOLID. In this case the CIM_StorageAllocationSettingData instances are not required to contain size information because the value of the HostResource[ ] array parameter directly identifies VOLID that is of a given size.

The lower part of Figure 35 shows the configuration of VOLID as it might be presented by an implementation of the SNIA SMIS Part 3 *Block Devices, Block Services* package. Several layered storage pools in scope of the host system enable the creation and management of block storage resources. The implementation of the *Extent Composition* subprofile on the right side enables the representation of cascaded combinations and / or subdivisions of storage extents.

Note that all aspects managed through SNIA SMIS profiles address the representation and management of *host* storage capacity and *host* storage elements. Opposed to that the main functionality specified by the profile described in this clause is the allocation and management of host resources in support of *virtual* storage resources such as *virtual* disks. In other words, the implementation of profiles from SNIA SMIS in combination with an implementation of the profile described in this clause is supplemental with respect to the representation and management of host storage resources, but not with respect to the allocation and management of virtual storage resources.

The advantage resulting from implementing profiles from SNIA SMIS along with implementing the profile described in this clause is that the implementation of profiles from SNIA SMIS enable a more granular management of host resources and storage pools. These host resources and storage pools may subsequently be referenced in instances of the CIM_StorageAllocationSettingData class that describe storage resource allocation requests and storage resource allocations as specified by the profile described in this clause.

10.2 Implementation

This subclause provides normative requirements related to the arrangement of instances and properties of instances for implementations of the profile described in this clause.

10.2.1 Common requirements

The CIM Schema descriptions for any referenced element and its sub-elements apply.

In references to properties of CIM classes that enumerate values the numeric value is normative and the descriptive text following it in parentheses is informative. For example, in the statement "The value of the ConsumerVisibility property shall be 3 (Virtualized)", the value "3" is normative text and "(Virtualized)" is informative text.

Implementations of the profile described in this clause shall expose an instance of the CIM_RegisteredProfile class as adapted in 8.5.9 in the Interop namespace. That instance shall be

associated with the CIM_RegisteredProfile instance representing the implementation of the scoping profile through an instance of the CIM_ReferencedProfile association as adapted in 10.5.10. Additional instance requirements specified in DSP1033 may apply.

10.2.2 Resource types

This subclause specifies the resource types that are addressed by the profile described in this clause.

10.2.2.1 General

The profile described in this clause may be implemented for the allocation of two principal resource types: *Storage extents* or *disk drives*. Note that *logical disks* and *storage volumes* are specializations of storage extents.

10.2.2.2 Logical disks, storage volumes and storage extents

This subclause provides definitions of the terms logical disk, storage volume and storage extent as well as their CIM representation as applied by the profile described in this clause. These definitions refine those provided in the CIM schema definitions of the CIM_LogicalDisk class, the CIM_StorageVolume class and the CIM_StorageExtent class and adopt the consistent parts of respective definitions provided in various places of SNIA SMIS for the purposes of the profile described in this clause.

NOTE The CIM schema definition of the CIM_LogicalDisk class, the CIM_StorageVolume class and the CIM_StorageExtent class as well as various subprofiles of SNIA SMIS present definitions of the terms logical disk, storage volume and storage extent. The essence of these definitions is that a storage extent is an abstraction of a range of storage media, that a logical disk is a consumed storage extent and that a storage volume is a storage extent exposed for external use by consumers.

A *storage extent* is a logically contiguous range of logical blocks on some storage media that supports storing and retrieving data. Storage extents shall be represented by CIM_StorageExtent instances, or by instances of subclasses of the CIM_StorageExtent class if the stricter definitions below apply.

A *logical disk* is a specialization of storage extent that is exposed by the virtualization platform to a virtual system for directly consumption. Logical disks shall be represented by CIM_LogicalDisk instances.

A *storage volume* is a specialization of storage extent that is exposed by the host or by a virtual storage array for complete or partitioned use by virtual systems. Storage volumes shall be represented by CIM_StorageVolume instances. This applies likewise to storage volumes exposed by the host or exposed by a virtual storage array.

10.2.2.3 Disk drives

A disk drive is a media access device; it provides the functionality to access some kind of media. Disk drives shall be represented by CIM_DiskDrive instances.

An implementation of the profile described in this clause for the allocation of storage extents may allocate disk drives as a side effect of the allocation of storage extents.

10.2.3 Host resources

This subclause specifies requirements for the representation of host resources.

10.2.3.1 Host storage volume

The representation of host storage volumes is conditional.

Condition: The profile described in this clause is implemented for one of the resource types 31 (Logical Disk), 32 (Storage Volume), or 19 (Storage Extent), and the resource aggregation feature (see 10.2.5) is implemented.

Each host storage volume that is a component of a storage resource pool shall be represented by exactly one CIM_StorageVolume instance as adapted in 10.5.15. The CIM_StorageVolume instance shall be associated with the CIM_System instance that represents the host system through an instance of the CIM_SystemDevice association, and with the CIM_ResourcePool instance representing the aggregating resource pool through an instance of a subclass of the CIM_Component association as adapted in 10.5.1.

10.2.3.2 Host disk drives

The representation of host disk drives is conditional.

Condition: The profile described in this clause is implemented for the resource type 17 (Disk Drive), and the resource aggregation feature (see 10.2.5) is implemented.

Each host disk drive volume that is a component of a storage resource pool shall be represented by exactly one CIM_DiskDrive instance as adapted in 10.5.2. The CIM_DiskDrive instance shall be associated with the CIM_System instance that represents the host system through an instance of the CIM_SystemDevice association, and with the CIM_ResourcePool instance representing the aggregating resource pool through an instance of a subclass of the CIM_Component association as adapted in 10.5.1.

10.2.4 Resource pools

This subclause adapts the CIM_ResourcePool class for the representation of storage resource pools and for disk drive resource pool.

10.2.4.1 General

Implementations of the profile described in this clause for one of the resource types 31 (Logical Disk), 32 (Storage Volume) or 19 (Storage Extent) may chose to implement the CIM_StoragePool class (which is a subclass of the CIM_ResourcePool class) in place of the CIM_ResourcePool class. The provisions in this subclause apply likewise to implementations of the CIM_ResourcePool class itself, or to implementations of the CIM_StoragePool class if that is implemented instead of the CIM_ResourcePool class. The profile described in this clause does not adapt properties defined by the CIM_StoragePool class.

NOTE The SNIA SMIS Part 3 Block Devices, Block Services package may be implemented for the management of host storage resources; see 10.1.4.10. Note that the adaptation of the CIM_StoragePool class in the SNIA SMIS Part 3 Block Devices, Block Services package imposes much stricter implementation requirements for the CIM_StoragePool class than the profile described in this clause.

10.2.4.2 ResourceType property

The value of the ResourceType property shall denote the type of resources that are provided by the resource pool, as follows:

- For resource pools supporting only the allocation of logical disks the value of the ResourceType property shall be 31 (Logical Disk).
- For resource pools supporting only the allocation of storage volumes the value of the ResourceType property shall be 32 (Storage Volume).
- For resource pools supporting the allocation of basic storage extents, logical disks or storage volumes the value of the ResourceType property shall be 19 (Storage Extent).

NOTE See 10.2.2.2 for a definition of logical disk, storage volume and storage extent.

- For resource pools supporting the allocation of disk drives the value of the ResourceType property shall be 17 (Disk Drive).

10.2.4.3 ResourceSubType property

The implementation of the ResourceSubType property is optional.

If the ResourceSubType property is implemented, the provisions in this subclause apply.

The value of the ResourceSubType property shall designate a resource subtype. The format of the value shall be as follows: "<org-id>:<org-specific>". The <org-id> part shall identify the organization that defined the resource subtype value; the <org-specific> part shall uniquely identify a resource subtype within the set of subtype defined by the respective organization.

EXPERIMENTAL NOTE: The following content is experimental.

An implementation may use the predefined values in Table 147. However implementations are not bound to apply these values; instead, implementation may apply other vendor defined values.

Table 147 – Predefined ResourceSubType values (EXPERIMENTAL)

ResourceSubType value	Description
"DMTF:generic:scsi"	Storage device that appears as SCSI device to the guest operating system
"DMTF:generic:ide"	Storage device that appears as IDE device to the guest operating system
"DMTF:generic:virtualblockdevice"	Storage device that appears as generic block device to the guest operating system. NOTE: This definition of block device is system virtualization specific; it does not imply properties of block devices as defined in SNIA SMIS Part 3 *Block Devices, Block Services*.
"DMTF:ibm:z:3380"	Storage device that appears as IBM 3380 device to the guest operating system
"DMTF:ibm:z:3390"	Storage device that appears as IBM 3390 device to the guest operating system
"DMTF:ibm:z:9336"	Storage device that appears as IBM 9336 device to the guest operating system
"DMTF:ibm:z:FB-512"	Storage device that appears as IBM FB-512 device to the guest operating system
"DMTF:xen:vbd"	Storage device that appears as Xen virtual block device to the guest operating system

The implementation should apply the mechanisms defined in the Allocation Capabilities Profile (described in clause 7) to expose the resource subtypes that are supported by the implementation.

10.2.4.4 Primordial property

The value of the Primordial property shall be TRUE in any instance of the CIM_ResourcePool class that represents a primordial resource pool. The value of the Primordial property shall be FALSE in any instance of the CIM_ResourcePool class representing a concrete resource pool.

NOTE See 10.1.3.3 and 10.1.3.4 for definitions of primordial and concrete resource pools.

10.2.4.5 PoolID property

The value of the PoolID property shall enable unique identification of the CIM_ResourcePool instance within the scoping host system.

10.2.4.6 Reserved property

The implementation of the Reserved property is optional.

If the Reserved property is implemented, its value shall denote the amount of resource that is actually allocated from the resource pool, as follows:

- If the value of the ResourceType property is any of 31 (Logical Disk), 32 (Storage Volume) or 19 (Storage Extent), the value of the Reserved property shall reflect the amount of storage that is allocated from the resource pool, in units as specified by the value of the AllocationUnits property (see 10.2.4.8).
- If the value of the ResourceType property is 17 (Disk Drive), the value of the Reserved property shall reflect the number of drives that is allocated from the resource pool.

NOTE For the resource type 17 (Disk Drive), the value of the AllocationUnits property is fixed to "count".

The special value NULL shall be used if the implementation does not have knowledge about the amount of resource allocated from the pool. This may reflect a permanent or a temporary situation.

10.2.4.7 Capacity property

The implementation of the Capacity property is conditional.

Condition: The resource aggregation feature is implemented; see 10.2.5.

If the Capacity property is implemented, its value shall reflect the maximum amount of resource that can be allocated from the resource pool, as follows:

- If the value of the ResourceType property is any of 31 (Logical Disk), 32 (Storage Volume) or 19 (Storage Extent), the value of the Capacity property shall reflect the maximum amount of storage that can be allocated from the resource pool, in units as specified by the value of the AllocationUnits property.
- If the value of the ResourceType property is 17 (Disk Drive), the value of the Capacity property shall reflect the maximum number of disk drives that can be allocated from the resource pool.

NOTE For the resource type 17 (Disk Drive), the value of the AllocationUnits property is fixed to "count".

The special value NULL shall be used if the implementation does not have knowledge about the resource capacity represented by the pool. This may reflect a permanent or a temporary situation.

10.2.4.8 AllocationUnits property

The value of the AllocationUnits property shall denote the unit of measurement that applies to resource allocations obtained from the resource pool:

- If the value of the ResourceType property is either 31 (Logical Disk), 32 (Storage Volume) or 19 (Storage Extent), the value of the AllocationUnit property shall express the minimum block size that is supported for the type of host storage extent represented by the resource pool. The value shall match "^byte(*([0-9]{1,}|(2|10)\^[0-9]{1,2})){0,1}$".

NOTE The regular expression specifies the basic unit "byte". In order to express a minimum block size the basic unit "byte" may be refined with a factor. The factor may be expressed as a plain number (such as "byte*4096"), or may be based on a power of either 2 (such as "byte*2^10" (kibibyte)) or 10 (such as "byte*10^3" (kilobyte)).

- If the value of the ResourceType property is 17 (Disk Drive), the value of the AllocationUnits property shall be "count".

10.2.4.9 MaxConsumableResource property

The implementation of the MaxConsumableResource property is optional.

If the MaxConsumableResource property is implemented, its value shall reflect the maximum amount of resource that is allocatable to consumers, in units as expressed by the value of the ConsumedResourceUnit property (see 10.2.4.11).

NOTE This property describes the consumer side of allocations, as opposed to the providing side that is described by the Capacity property. This allows the representation of resource pools that support over-commitment. For example, a resource pool of the type 31 (Logical Disk) might be able to support virtual disks with an added up virtual quantity of 4 GB, and base that on a file system capacity of 2 GB.

10.2.4.10 CurrentlyConsumedResource property

The implementation of the CurrentlyConsumedResource property is optional.

If the CurrentlyConsumedResource property is implemented, its value shall reflect the actually allocated amount of resource to consumers, in units as expressed by the value of the ConsumedResourceUnit property (see 10.2.4.11).

10.2.4.11 ConsumedResourceUnit property

The implementation of the ConsumedResourceUnit property is conditional.

Condition: The MaxComsumableResource property (see 10.2.4.9) or the CurrentlyConsumedResource property (see 10.2.4.10), or both, are implemented.

If the CurrentlyConsumedResource property is implemented, its value shall state the unit that applies to the values of the MaxComsumableResource property and the CurrentlyConsumedResource property; the same rules as for the AllocationUnits property (see 10.2.4.8) apply.

10.2.4.12 Instance requirements

Each resource pool shall be represented by a CIM_ResourcePool instance; the provisions of 10.5.13 apply.

10.2.5 Resource aggregation feature

The implementation of the resource aggregation feature is conditional.

Condition: The resource pool management feature is implemented; see 10.2.7.

Granularity: If implemented, the resource aggregation feature may be separately supported for each resource pool.

The preferred feature discovery mechanism is to resolve the CIM_Component association from the CIM_ResourcePool instance to CIM_ManagedElement instances representing aggregated resources of the storage resource pool. If the resulting set of CIM_ManagedElement instances is not empty, the feature is supported.

NOTE If the result set is empty, the feature may still be supported, but no resources are aggregated at that point in time; however, if ever for a particular resource pool aggregated resources were exposed, then the feature is still supported even if at a later point in time no resources are aggregated.

10.2.6 Resource pool hierarchies feature

The implementation of the representation of resource pool hierarchies is optional.

Granularity: If implemented, the resource pool hierarchies feature may be separately supported for each resource pool.

If the representation of resource pool hierarchies is implemented, any concrete resource pool shall be represented through an instance of the CIM_ResourcePool class, where all of the following conditions shall be met:

- The value of the Primordial property shall be FALSE.
- The instance shall be associated through an instance of CIM_ElementAllocatedFromPool association to the instance of the CIM_ResourcePool class that represents its parent resource pool.

NOTE The SNIA SMIS Part 3 *Block Devices, Block Services* package requires the implementation of the CIM_ConcreteComponent association for the representation of the same relationship if the SNIA SMIS *Part 3 Block Devices, Extent Composition* subprofile is implemented; in this case implementations of the SNIA SMIS: Part 3 *Block Devices, Block Services* package need to implement both associations.

- The instance shall be associated through an instance of the CIM_ElementSettingData association to the instance of the CIM_ResourceAllocationSettingData class that represents the amount of resource allocated from the parent pool.

The preferred feature discovery mechanism is to resolve the CIM_ElementAllocatedFromPool association from a CIM_ResourcePool instance to other (parent or child) CIM_ResourcePool instances. If the resulting set of CIM_ResourcePool instances is not empty, the feature is supported; otherwise, the feature is not supported.

NOTE If for example the Associators( ) intrinsic operation is used to resolve the association, the Role parameter or the ResultRole parameters may be used to distinguish the parent-to-child relationship from the child-to-parent relationship.

10.2.7 Resource pool management feature

The implementation of the resource pool management feature is optional.

If implemented, the specifications of the Resource Allocation Profile described in clause 5 apply; the profile described in this clause does not specify specializations or extensions of resource pool management beyond those defined by the Resource Allocation Profile (see clause 5).

10.2.8 Resource allocation

This subclause details requirements for the representation of resource allocation information through CIM_ResourceAllocationSettingData (RASD) instances or CIM_StorageAllocationSettingData (SASD) instances.

10.2.8.1 General

Implementations of the profile described in this clause shall implement the virtual resource allocation pattern as defined in 5.2.2.

NOTE The Resource Allocation Profile described in clause 5 specifies two alternatives for modeling resource allocation: *simple resource allocation* and *virtual resource allocation*.

The profile described in this clause adapts the CIM_StorageAllocationSettingData (SASD) class for storage resource allocation and the CIM_ResourceAllocationSettingData class for disk drive resource allocation.

10.2.8.2 Flavors of allocation data

Various flavors of allocation data describes are defined:

- Resource allocation requests; for details see 10.1.4.3.
- Resource allocations; for details see 10.1.4.4.
- Settings that define the capabilities or mutability of managed resources. The Allocation Capabilities Profile described in clause 7 specifies a capabilities model that conveys information about the capabilities and the mutability of managed resources in terms of RASD instances.
- Parameters in operations that define or modify any of the representations listed above. The System Virtualization Profile (see clause 6) that specifies methods for the definition and modification of virtual resources. These methods use RASD instances for the parameterization of resource-allocation-specific properties.

Table 148 lists acronyms that are used in subclauses of 10.2.8 in order to designate RASD or SASD instances that represent various flavors of allocation data.

Table 148 – Acronyms for RASD adapted for the representation of various flavors of allocation data

Acronym	Flavor
Q_RASD	RASD adapted for the representation of disk drive resource allocation requests
Q_SASD	SASD adapted for the representation of storage resource allocation requests
R_RASD	RASD adapted for the representation of disk drive resource allocations
R_SASD	SASD adapted for the representation of storage resource allocations
C_RASD	RASD adapted for the representation of settings that define capabilities of systems or disk drive resource pools, or that define the mutability of disk drive allocations or disk drive allocation requests
C_SASD	SASD adapted for the representation of settings that define capabilities of systems or storage resource pools, or that define the mutability of storage resource allocations or storage resource allocation requests
D_RASD	RASD adapted for the representation of new disk drive resource allocation requests in method parameter values
D_SASD	SASD adapted for the representation of new storage resource allocation requests in method parameter values
M_RASD	RASD adapted for the representation of modified disk drive resource allocations or disk drive resource allocation request in method parameter values
M_SASD	SASD adapted for the representation of modified storage resource allocations or storage resource allocation request in method parameter values

Subclauses of 10.2.8 detail implementation requirements for property values in RASD instances. In some cases requirements only apply to a subset of the flavors listed in Table 148; this is marked in the text through the use of respective acronyms.

10.2.8.3 CIM_ResourceAllocationSettingData properties

This subclause defines rules for values of properties in instances of the CIM_ResourceAllocationSettingData (RASD) class representing disk drive allocation information.

10.2.8.3.1 ResourceType property

The value of the ResourceType property in RASD instances representing disk drive allocation information shall be set to 17 (Disk Drive) for disk drive allocation data.

Other values shall not be used.

10.2.8.3.2 ResourceSubType property

The implementation of the ResourceSubType property is optional.

If the ResourceSubType property is implemented, the provisions defined for the ResourceSubType property of the CIM_ResourcePool class; see 10.2.4.2.

10.2.8.3.3 PoolID property

The value of the PoolID property shall identify the resource pool. The special value NULL shall indicate the use of the host system's default resource pool for the selected resource type.

10.2.8.3.4 ConsumerVisibility property

The implementation of the ConsumerVisibility property is optional.

If the ConsumerVisibility property is implemented, the provisions in this subclause apply.

The value of the ConsumerVisibility property shall denote either if a host resources is directly passed through to the virtual system as a virtual resource, or if the resource is virtualized. Values shall be set as follows:

- A value of 2 (Passed-Through) shall denote that the host resource is passed-through.
- A value of 3 (Virtualized) shall denote that the virtual resource is virtualized.
- Only in instances of { Q_RASD | D_RASD | M_RASD }, the special value NULL shall be used if the represented resource allocation request does not predefine which kind of consumer visibility (passed-through or virtualized) is requested.
- Other values shall not be used.

10.2.8.3.5 HostResource[ ] array property

The implementation of the HostResource[ ] array property is conditional.

Condition: The HostResource[ ] array property shall be implemented if any of the following conditions is true: The value 2 (Passed-Through) is supported for the value of the ConsumerVisibility property, or any of the values 3 (Dedicated), 4 (Soft Affinity) or 5 (Hard Affinity) is supported for the MappingBehavior property.

If the HostResource[ ] array property is implemented, the provisions in this subclause apply.

In the cases of { Q_RASD | C_RASD | D_RASD | M_RASD } the value of the HostResource[ ] array property shall refer to the representation of one or more host resources that are configured to contribute to the disk drive resource allocation. In the case of R_RASD the value of the HostResource[ ] array property shall refer to a representation of the host resource that provides the disk drive resource allocation.

Elements of the value of the HostResource[ ] array property shall refer to instances of CIM classes, using the WBEM URI format as specified by DSP0207. Referenced instances shall be of the CIM_CDROMDrive class, the CIM_DiskDrive class, the CIM_DisketteDrive class, the CIM_DVDDrive class or the CIMWORMDrive class.

10.2.8.3.6 AllocationUnits property

The value of the AllocationUnits property shall be "count".

NOTE The units defined by value of the AllocationUnits property applies to the values of the Reserved and the Limit property; it does not apply to the value of the VirtualQuantity property.

10.2.8.3.7 VirtualQuantity property

The value of the VirtualQuantity property shall denote the number of virtual disk drives available to a virtual system through this resource allocation.

10.2.8.3.8 VirtualQuantityUnits property

EXPERIMENTAL NOTE: The following content is experimental.

The value of the VirtualQuantityUnits property shall be "count".

10.2.8.3.9 Reservation property

The implementation of the Reservation property is optional.

If the Reservation property is implemented, the provisions in this subclause apply.

The value of the Reservation property shall denote the number of disk drives reserved through this resource allocation to a virtual system.

10.2.8.3.10 Limit property

The implementation of the Limit property is optional.

If the Limit property is implemented, the following rules apply:

The value of the Limit property shall denote the maximum number of disk drives available through this resource allocation to a virtual system.

10.2.8.3.11 Weight property

The implementation of the Weight property is optional.

If the Weight property is implemented, its value shall denote the relative priority of a resource allocation in relation to other resource allocations.

10.2.8.3.12 Parent property

The implementation of the Parent property is optional.

If the Parent property is implemented, the provisions in this subclause apply.

The value of the Parent property shall identify the parent entity of the resource allocation or resource allocation request. The value of the Parent property shall be formatted with the WBEM URI format as specified by DSP0207.

If an implementation implements the concept of disk snapshots where data stored on a delta disk only contains information on top of that stored on a base disk, then the implementation should use the value of

the Parent property in the RASD instance representing the storage resource allocation of the delta disk to refer to the RASD instance representing the storage resource allocation of the base disk.

10.2.8.3.13 Connection[ ] array property

The implementation of the Connection[ ] array property is optional.

If the Connection[ ] array property is implemented, the provisions in this subclause apply.

The value of the connection property may identify elements of the storage infrastructure such as initiator ports and/or target ports. The WBEM URI format (see DSP0207) may be used to refer to a respective CIM instance.

10.2.8.3.14 Address property

The implementation of the Address property is optional.

If the Address property is implemented, the provisions in this subclause apply.

The value of the Address property shall expose the address of the allocated resource as seen by the software running in the virtual system (usually a guest operating system).

10.2.8.3.15 MappingBehavior property

The implementation of the MappingBehavior property is optional.

If the MappingBehavior property is implemented, its value shall denote how host resources referenced by elements in the value of HostResource[ ] array property relate to the resource allocation.

In R_RASD instances the following rules apply to the value of the MappingBehavior property:

- A value of 2 (Dedicated) shall indicate that the represented resource allocation is provided by host resources that are exclusively dedicated to the virtual system. The host resources shall be identified by the value of the HostResource[ ] array property.
- A value of 3 (Soft Affinity) or 4 (Hard Affinity) shall indicate that the represented resource allocation is provided by host resources. The host resources shall be identified by the value of the HostResource[ ] array property.
- Other values shall not be used.

In Q_RASD instances the following rules apply to the value of the MappingBehavior property:

- The special value NULL shall indicate that the resource allocation request does not require specific host resources.
- A value of 2 (Dedicated) shall indicate that the resource allocation request shall be provided by exclusively dedicated host resources as specified through the value of the HostResource[ ] array property.
- A value of 3 (Soft Affinity) shall indicate that the resource allocation request shall preferably be provided by host resources as specified through the value of the HostResource[ ] array property, but that other host resources may be used if the requested host resources are not available.
- A value of 4 (Hard Affinity) shall indicate that the resource allocation request shall be provided by host resources as specified through the value of the HostResource[ ] array property and that other resources shall not be used if the requested host resources are not available.
- Other values shall not be used.

10.2.8.4 CIM_StorageAllocationSettingData properties

This subclause defines rules for values of properties in instances of the CIM_StorageAllocationSettingData (SASD) class.

NOTE If the rules for a particular property are the same as those defined for the respective property of the CIM_ResourceAllocationSettingData (RASD) class, the respective subclause of 10.2.8.3 is referenced.

10.2.8.4.1 ResourceType property

The value of the ResourceType property shall be set as follows:

- 31 (Logical Disk) for logical disk allocation data
- 32 (Storage Volume) for storage volume allocation data
- 19 (Storage Extent) for storage extent allocation data

NOTE See 10.2.2.2 for a definition of logical disk, storage volume and storage extent.

10.2.8.4.2 ResourceSubtype property

See 10.2.8.3.2.

10.2.8.4.3 PoolID property

See 10.2.8.3.3.

10.2.8.4.4 ConsumerVisibility property

See 10.2.8.3.4.

10.2.8.4.5 HostResource array property

The implementation of the HostResource[ ] array property is conditional.

Condition: The HostResource[ ] array property shall be implemented if any of the following conditions is true: The value 2 (Passed-Through) is supported for the value of the ConsumerVisibility property, or any of the values 3 (Dedicated), 4 (Soft Affinity) or 5 (Hard Affinity) is supported for the MappingBehavior property.

If the HostResource[ ] array property is implemented, the provisions in this subclause apply.

In the cases of { Q_SASD | C_SASD | D_SASD | M_SASD } the value of the HostResource[ ] array property shall refer to (the representation of) one or more host resources that are configured to contribute to the resource allocation. In the case of R_SASD the value of the HostResource[ ] array property shall refer to (the representation of) the host resource that provides the storage resource allocation.

Values of elements of the HostResource[ ] array property may directly refer to files, using the URI format as specified by IETF RFC1738 and file URL scheme as specified in IETF RFC3986.

If the file URI is not applied, elements of the value of the HostResource[ ] array property shall refer to instances of CIM classes, using the WBEM URI format as specified by DSP0207. Referenced instances shall be of the CIM_StorageExtent class or the CIM_LogicalFile class.

10.2.8.4.6 AllocationUnits property

The implementation of the AllocationUnits property is conditional.

Condition: The Reservation property (see 10.2.8.4.9) or the Limit property (see 10.2.8.4.10), or both are implemented.

The AllocationUnits property shall convey the unit applicable to the values of the Reservation and the Limit property.

If the AllocationUnits property is implemented, the provisions in this subclause apply.

If the value of the BlockSize property is 1, the value of the AllocationUnits property shall be "byte", indicating the that the values of the Reservation and of the Limit property are specified in bytes. If the value of the BlockSize property is greater than 1, the value of the AllocationUnits property shall be "count", indicating that the values of the Reservation and of the Limit property are specified in blocks, with the blocksize conveyed through the value of the BlockSize property.

All flavors of SASD instances as defined in 10.2.8.2 that relate to the same virtual resource shall apply the same value for the AllocationUnits property.

NOTE The definitions in this subclause include SASD instances that describe mutability. In these instances the mutability is expressed by values of numerical properties such as the Reservation or the Limit property in units as established by the value of the AllocationUnit property. If the mutability SASD instance represents an increment, this would reflect a granularity for modifications of the numeric property values that is equal to or a multiple of the allocation unit.

10.2.8.4.7 VirtualQuantity property

In the cases of { R_SASD | C_SASD | D_SASD | M_SASD } the value of the VirtualQuantity property shall denote the amount of storage that is available to a virtual system through this resource allocation. In the case of Q_SASD the value of the VirtualQuantity property shall denote the amount of storage that is requested for the virtual system unless the value of the HostResource[ ] array property contains exactly one element that refers to a specific host storage resource that implicitly determines the virtual disk size. If a value is provided, is shall be expressed in units as expressed by the value of the VirtualQuantityUnit property; see 10.2.8.4.8.

10.2.8.4.8 VirtualQuantityUnits property

EXPERIMENTAL NOTE: The following content is experimental.

The VirtualQuantityUnits property shall convey the unit applicable to the value of the VirtualQuantity property.

If the value of the VirtualResourceBlockSize property is 1, the value of the VirtualQuantityUnits property shall be "byte", indicating the that the value of the VirtualQuantity property is specified in bytes. If the value of the VirtualBlockSize property is greater than 1, the value of the VirtualQuantityUnits property shall be "count", indicating that the value of the VirtualQuantity property is specified in blocks, with the blocksize conveyed through the value of the VirtualResourceBlockSize property.

10.2.8.4.9 Reservation property

The implemenation of the Reservation property is optional.

If the Reservation property is implemented, the provisions in this subclause apply.

The value of the Reservation property shall denote the amount of storage reserved through this resource allocation to a virtual system in units as expressed by the value of the AllocationUnits property; see 10.2.8.4.6.

10.2.8.4.10 Limit property

The implementation of the Limit property is optional.

If the Limit property is implemented, the provisions in this subclause apply.

The value of the Limit property shall denote the maximum amount of storage available through the represented resource allocation to a virtual system in units as expressed by the value of the AllocationUnits property; see 10.2.8.4.6.

10.2.8.4.11 Weight property

See 10.2.8.3.11.

10.2.8.4.12 Parent property

See 10.2.8.3.12.

10.2.8.4.13 Connection[ ] array property

See 10.2.8.3.13.

10.2.8.4.14 Address property

See 10.2.8.3.14.

10.2.8.4.15 MappingBehavior property

See 10.2.8.3.15.

10.2.8.4.16 VirtualResourceBlockSize

The value of the VirtualResourceBlockSize property shall denote the block size as seen by the consumer of a virtual storage that is based on the described resource allocation. A value of 1 shall designate a variable block size.

10.2.8.4.17 Access

The value of the Access property shall denote the access mode.

10.2.8.4.18 HostResourceBlockSize

The value of the HostResourceBlockSize property shall denote the block size as seen by the consumer of a virtual storage that is based on the described resource allocation. A value of 1 shall designate a variable block size.

10.2.8.4.19 HostExtentStartingAddress

The implementation of the HostExtentStartingAddress property is optional.

If the HostExtentStartingAddress property is implemented, the provisions in this subclause apply.

The value of the HostExtentStartingAddress property shall denote the offset within the host storage extent referenced by the value of the HostExtentName property. The offset marks the starting point of a subspace within the referenced host storage extent. The size of the subspace is exposed by the value of the Reserved property.

10.2.8.4.20 HostExtentName

The implementation of the HostExtentName property is optional.

If the HostExtentName property is implemented, the provisions in this subclause apply.

The value of the HostExtentName shall identify a host storage extent that serves as the base for the described virtual storage allocation.

10.2.8.4.21 HostExtentNameFormat

The implementation of the HostExtentNameFormat property is conditional.

Condition: The HostExtentName property (see 10.2.8.4.20) is implemented.

If the HostExtentNameFormat property is implemented, the provisions in this subclause apply.

The value of the HostExtentNameFormat shall designate the format used for the value of the HostExtentName property.

10.2.8.4.22 OtherHostExtentNameFormat

The implementation of the HostExtentNameFormat property is conditional.

Condition: The HostExtentNameFormat property (see 10.2.8.4.21) is implemented, and the value 1 (Other) is supported.

If the OtherHostExtentNameFormat property is implemented, the provisions in this subclause apply.

The value of the HostExtentNameFormat shall designate the format used for the value of the HostExtentName property, using a string representation. The value should be structured as follows: <Organization>:<FormatSpecifier>. <Organization> shall uniquely identify the organization that defined the format. <FormatSpecifier> shall uniquely identify the format within the set of formats defined by the organization.

10.2.8.4.23 HostExtentNameNamespace

The implementation of the HostExtentNameNamespace property is conditional.

Condition: The HostExtentName property (see 10.2.8.4.20) is implemented.

If the HostExtentNameNamespace property is implemented, the provisions in this subclause apply.

The value of the HostExtentNameNamespace shall designate the namespace that applies to the value of the HostExtentName property.

10.2.8.4.24 OtherHostExtentNameNamespace

The implementation of the OtherHostExtentNameNamespace property is conditional.

Condition: The HostExtentNameNamespace property (see 10.2.8.4.23) is implemented, and the value 1 (Other) is supported.

If the OtherHostExtentNameNamespace property is implemented, the provisions in this subclause apply.

The value of the HostExtentNameNamespace shall designate the namespace used for the value of the HostExtentName property, using a string representation. The value should be structured as follows: <Organization>:<NamespaceSpecifier>. <Organization> shall uniquely identify the organization that defined the format. <NamespaceSpecifier> shall uniquely identify the namespace within the set of namespaces defined by the organization.

10.2.8.5 Instance requirements

This subclause details resource allocation related instance requirements.

10.2.8.5.1 Representation of resource allocation requests

If the profile described in this clause is implemented for the allocation of storage extents (see 10.2.2), each storage resource allocation request shall be represented by a Q_SASD instance; the provisions of 10.5.15 apply.

If the profile described in this clause is implemented for the allocation of disk drives (see 10.2.2), each disk drive resource allocation request shall be represented by a Q_RASD instance; the provisions of 10.5.12 apply.

10.2.8.5.2 Representation of resource allocations

If the profile described in this clause is implemented for the allocation of storage extents (see 10.2.2), each storage resource allocation shall be represented by a R_SASD instance; the provisions of 10.5.15 apply.

If the profile described in this clause is implemented for the allocation of disk drives (see 10.2.2), each disk drive resource allocation shall be represented by a R_RASD instance; the provisions of 10.5.12 apply.

The R_SASD (or R_RASD) instance shall be associated to the Q_SASD (or Q_RASD) instance representing the corresponding resource allocation request (see 10.2.8.5.1) through an instance of the CIM_ElementSettingData association; the provisions of 0 apply.

The R_SASD (or R_RASD) instance shall be associated to the CIM_ResourcePool instance providing resources for the allocation (see 10.2.4) through an instance of the CIM_ResourceAllocationFromPool association; the provisions of 0 apply.

Implementations may represent a resource allocation request and the corresponding resource allocation by one SASD (or RASD) instance; in this case the association requirements of this subclause apply correspondingly. Note that association instances that refer to the R_SASD instance are only existent while the resource is allocated.

10.2.8.5.3 Representation of resource allocation capabilities

The allocation capabilities of a system or a resource pool shall be represented by an CIM_AllocationCapabilities instance that is associated to the CIM_System instance representing the system or to the CIM_ResourcePool instance representing the resource pool through an instance of the CIM_ElementCapabilities association; see the Allocation Capabilities Profile described in clause 7.

The settings that define the allocation capabilities of a storage resource pool shall be represented by C_SASD instances; the provisions of 10.5.15 apply.

The settings that define the allocation capabilities of a disk drive resource pool shall be represented by C_RASD instances; the provisions of 10.5.12 apply.

10.2.8.5.4 Representation of resource allocation mutability

The mutability of a resource allocation or resource allocation request shall be represented by an CIM_AllocationCapabilities instance that is associated to the RASD instance representing the resource allocation of resource allocation request through an instance of the CIM_ElementCapabilities association; see the Allocation Capabilities Profile described in clause 7.

The settings that define the allocation capabilities of a storage resource pool shall be represented by C_SASD instances; the provisions of 10.5.15 apply.

The settings that define the allocation capabilities of a disk drive resource pool shall be represented by C_RASD instances; the provisions of 10.5.12 apply.

10.2.9 Virtual resources

This subclause specifies rules for the representation of virtual resources. Virtual resources are the result of resource allocations. Virtual resources are scoped by virtual systems or by virtual storage arrays. Virtual storage arrays are a special kind of virtual system that serve the purpose of providing storage to other virtual systems.

10.2.9.1 Virtual resource instance requirements

An allocated virtual resource shall be represented by an instance of a subclass of the CIM_LogicalDevice class, as follows:

- Virtual disks

  The representation of virtual disks is governed by the type of virtual disk as defined in 10.2.2.2.

  – An allocated virtual disk shall be represented by a CIM_StorageExtent instance if the value of the ResourceType property in the CIM_StorageAllocationSettingData instance representing the virtual disk allocation is 19 (Storage Extent). However, if the allocated virtual disk conforms to the stricter definitions of logical disk or storage volume (see 10.2.2.2), it may be represented by a CIM_LogicalDisk or a CIM_StorageVolume, respectively.

  – A allocated virtual disk shall be represented by a CIM_StorageVolume instance if the value of the ResourceType property in the CIM_StorageAllocationSettingData instance representing the virtual disk allocation is 32 (Storage Volume)

  – A allocated virtual disk shall be represented by a CIM_LogicalDisk instance if the value of the ResourceType property in the CIM_StorageAllocationSettingData instance representing the virtual disk allocation is 31 (Logical Disk).

- Virtual disk drives

  A virtual disk drive shall be represented by an instance of the CIM_DiskDrive class

- Virtual ports

  If implemented, virtual initiator ports or virtual target ports shall be represented by instances of the CIM_LogicalPort class

Each instance of a subclass of the CIM_LogicalDevice class representing a virtual resource as defined in this subclause shall be associated as follows:

- to the CIM_ComputerSystem instance that represents the virtual system through an instance of the CIM_SystemDevice association
- to the SASD or RASD instance that represents the resource allocation through an instance of the CIM_SettingsDefineState association

  NOTE If the resource allocation of a logical disk is a composed resource allocation, only the top-most resource allocation yields a logical device. Consequently there may be SASD instances within a "state" virtual system configuration that do not have a companion logical device.

- to the CIM_ResourcePool instance that represents the resource pool providing the resource allocation through an instance of the CIM_ElementAllocatedFromPool association.

10.2.9.2 CIM_StorageExtent properties

This subclause defines constraints for property values in CIM_StorageExtent instances representing virtual disks.

10.2.9.2.1 BlockSize

The value of the BlockSize property shall be identical to the value of the VirtualResourceBlockSize property in the SASD instance representing the related storage resource allocation (see 10.2.8.4.16).

10.2.9.2.2 NumberOfBlocks

The value of the NumberOfBlocks property shall be identical to the value of the VirtualQuantity property in the SASD instance representing the related storage resource allocation (see 10.2.8.4.7).

10.2.9.2.3 Name

The value of the Name property shall expose the name of the storage extent as seen by the virtual system.

10.2.9.2.4 NameFormat

The value of the NameFormat property shall be 12 (OS Device Name).

10.2.9.2.5 NameNamespace

The value of the NameFormt property shall be 8 (OS Device Namespace).

10.3 Methods

This subclause details the requirements for supporting intrinsic operations and extrinsic methods for the CIM elements defined by the profile described in this clause.

10.3.1 Profile conventions for operations

The implementation requirements on intrinsic operations for each profile class (including associations) are specified in class-specific subclauses of this clause.

The default list of intrinsic operations for all classes is:

- GetInstance( )
- EnumerateInstances( )
- EnumerateInstanceNames( )

For classes that are referenced by an association, the default list also includes

- Associators( )
- AssociatorNames( )
- References( )
- ReferenceNames( )

Implementation requirements on operations defined in the default list are provided in the class-specific subclauses of this subclause.

The implementation requirements for intrinsic and extrinsic methods of classes listed in 8.5, but not addressed by a separate subclause of this subclause, are specified by the "Methods" clauses of respective base profiles, namely the Resource Allocation Profile described in clause 5 and the Allocation Capabilities Profile described in clause 7. These profiles are specialized by the profile described in this clause, and in these cases the profile described in this clause does not add method specifications beyond those defined in its base profiles.

10.3.2 CIM_DiskDrive for host disk drives

All intrinsic operations in the default list in 10.3.1 shall be implemented as specified by DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

10.3.3 CIM_DiskDrive for virtual disk drives

All intrinsic operations in the default list in 10.3.1 shall be implemented as specified by DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

10.3.4 CIM_LogicalDisk for virtual disk drives

All intrinsic operations in the default list in 10.3.1 shall be implemented as specified by DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

10.3.5 CIM_ReferencedProfile

All intrinsic operations in the default list in 10.3.1 shall be implemented as specified by DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

10.3.6 CIM_RegisteredProfile

All intrinsic operations in the default list in 10.3.1 shall be implemented as specified by DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

10.3.7 CIM_StorageAllocationSettingData for storage extent allocation information

All intrinsic operations in the default list in 10.3.1 shall be implemented as specified by DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

10.3.8 CIM_StorageExtent for virtual disk

All intrinsic operations in the default list in 10.3.1 shall be implemented as specified by DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

10.3.9 CIM_SystemDevice for host storage volumes

All intrinsic operations in the default list in 10.3.1 shall be implemented as specified by DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

10.3.10 CIM_SystemDevice for virtual resources

All intrinsic operations in the default list in 10.3.1 shall be implemented as specified by DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

10.4 Use cases

This subclause contains informative text only.

The following use cases and object diagrams illustrate use of the profile described in this clause. They are for informative purposes only and do not introduce behavioral requirements for implementations of the profile.

10.4.1 Instance diagram

Figure 36 depicts the CIM representation of a host system with one storage resource pool and one virtual system. Only information relevant in the context of storage resource virtualization is shown.

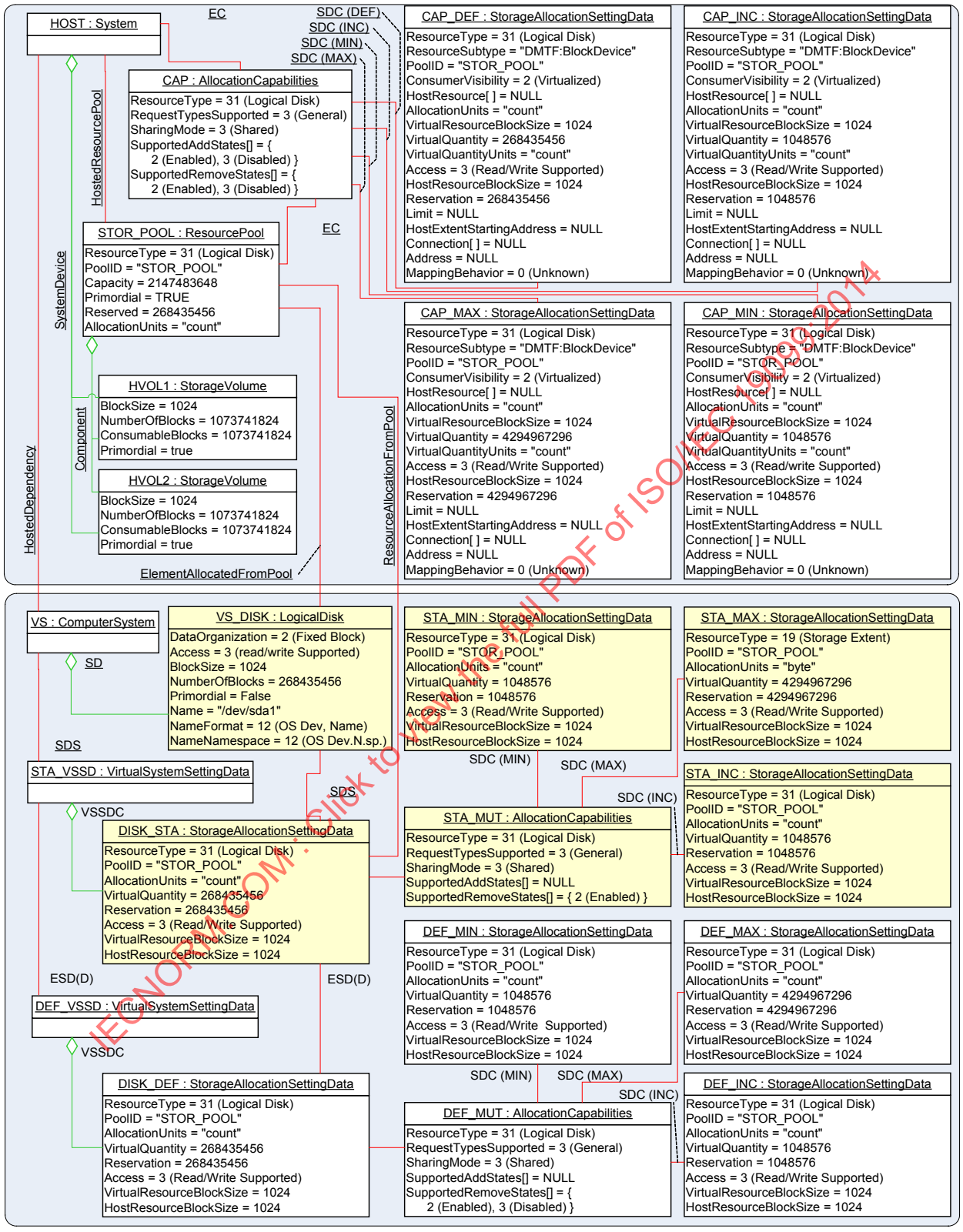

Figure 36 – Instance diagram: Example CIM representation of storage resource virtualization

In Figure 36 the host system is represented by an instance HOST of the CIM_System class. The host system owns or has access to two storage volumes each with a size of 1TB that are represented by the CIM_StorageExtent instances HVOL1 and HVOL2. Note that the storage volumes may be located within a storage area network that is not part of the host system itself.

The host system hosts a primordial storage resource pool that is represented by the CIM_ResourcePool instance STOR_POOL. The value of the ResourceType property in STOR_POOL is 31 (Logical Disk), designating the type of resources that are allocated out of the resource pool.

The resource type of resources aggregated by a resource pool may be different from the type of resources allocated from the pool. In this example as shown in Figure 36 both host storage volumes are aggregated into the pool, as represented by CIM_Component instances connecting STOR_POOL with HVOL1 and HVOL2, respectively.

In the example shown in Figure 36 the storage allocation capabilities of the host system and of the storage resource pool are identical and represented by the same CIM_AllocationCapabilities instance CAP. Four SASD instances (CAP_DEF, CAP_MIN, CAP_MAX, and CAP_INC) are associated with CAP through CIM_SettingsDefineCapabilities (SDC) instances. Not shown Figure 36 are the values of the ValueRange and ValueRole properties in the SDS instances that designate the referenced SASD instances as representing the default, minimum, maximum, and increment for storage resource allocations that are supported by the system and the pool; instead the lines depicting the association instances are respectively labeled as SDC(DEF, SDC(MIN), SDC(INC) and (SDC(MAX). The values of the VirtualQuantity property in the CAP_xxx instances indicate that virtual disks allocatable from the resource pool have a minimum supported size of 1 MB up to a maximum supported size of 4TB, and that an increment of 1 MB applies within the supported range; the default size is 256 GB.

In the CAP_xxx instances the value of the AllocationUnit property is "count"; this indicates that the value of the Reservation property is expressed in blocks. The block size is exposed by the value of the HostResourceBlockSize property (1024). Consequently storage allocations as seen from the providing host system or resource pool side are expressed in 1-KB blocks.

Similarly, the value of the VirtualQuantityUnits property is "count", indicating that the value of the VirtualQuantity property is expressed in blocks. Here the block size is exposed by the value of the VirtualResourceBlockSize property (1024). Consequently storage allocations as seen by the consuming virtual system are expressed in 1-KB blocks as well.

Note that the CAP_xxx instances do not expose a value for the Limit property. This indicates that the implementation does not support thin provisioning where the resource on the consuming side appears larger than the amount of resource provided at the providing side. This implies that the values of the numeric properties Reservation and VirtualQuantity are always identical for any resource allocation out of the resource pool.

The host system hosts a virtual system that is represented by the CIM_ComputerSystem instance VS. The hosted relationship is shown through a CIM_HostedDependency instance.

The head element of the "state" virtual system configuration is the VSSD instance STA_VSSD; it is associated with VS through a CIM_SettingsDefineState (SDS) instance. The "State" virtual system configuration contains the SASD instance DISK_STA that represents a storage resource allocation assigned to the virtual system. The virtual disk that is the result of the storage resource allocation is represented as part of the virtual system representation by the CIM_LogicalDisk instance VS_DISK.

NOTE All instances in Figure 36 that are marked with light yellow color represent "State" entities that exist only as long as the virtual system is instantiated (that is, in a state other than "Defined"). These instances do not exist while the virtual system is not instantiated (that is, in the "Defined" state).

The head element of the "defined" virtual system configuration is the VSSD instance DEF_VSSD; it is associated with the head element of the "state" virtual system configuration through an instance of the CIM_ElementSettingData association where the value of the IsDefault property is 1 (Is Default) (abbreviated as ESD(D) in Figure 36). The "defined" virtual system configuration contains the SASD

instance DISK_DEF that represents the respective storage resource allocation request. When the virtual system is activated, respective storage resources are allocated based on their definition.

Similarly to the representation of the allocation capabilities of a resource pool or system, the mutability of both the storage resource allocation request in the "Defined" virtual system configuration and of the storage resource allocation in the "State" virtual system configuration is represented by CIM_AllocationCapabilities instances with associated SASD instances through parameterized CIM_SettingsDefineCapabilities instances designating the minimum, maximum, and increment for storage resource allocation changes.

Acceptable virtual system states for the removal of virtual disks are different for the storage resource allocation request and the storage resource allocation. The storage resource allocation can be removed only while the virtual system remains instantiated, as indicated by a value of 2 (Enabled) in the CIM_AllocationCapabilities instance STA_MUT. This is a manifestation of the previously mentioned fact that the "state" configuration is not present while the virtual system is in the "defined" state.

10.4.2 Inspection

This set of use cases describes how to obtain various CIM instances that represent storage-related information of host and virtual systems.

10.4.2.1 Inspect the set of virtual disks of an active virtual system

10.4.2.1.1 Preconditions

All of the following:

- The client knows a reference to the CIM_ComputerSystem instance that represents the active virtual system.

10.4.2.1.2 Flow of activities

1) From the CIM_ComputerSystem instance the client resolves the CIM_SystemDevice association to find the CIM_LogicalDisk instances that represent virtual disks.
2) For each element of the result set of step 1) the client applies the use case in 10.4.2.2.

10.4.2.1.3 Postconditions

The client knows the virtual disks of the virtual system and their properties.

10.4.2.2 Inspect the properties of a virtual disk

10.4.2.2.1 Preconditions

All of the following:

- The client knows a reference to the instance of the CIM_LogicalDisk class that represents the virtual disk.

10.4.2.2.2 Flow of activities

The client obtains the CIM_LogicalDisk instance, using the GetInstance( ) intrinsic operation. In that instance, the client interprets property values such as the following:

- The value of the BlockSize property conveys the block size in effect for the virtual disks
- The value of the NumberOfBlocks property conveys the size of the virtual disk as seen by the virtual system as a number of blocks

- The value of the Name property conveys the name of the virtual disk as seen by the virtual system

10.4.2.2.3 Postconditions

The client knows properties of the virtual disk.

10.4.2.3 Determine the allocation capabilities or allocation mutability

This use case is applicable in two cases:

- Case (A) – Determine the capabilities of a system or a resource pool: In this case the entry element is the CIM_System instance representing the host system or the CIM_ResourcePool instance representing the resource pool.
- Case (B) – Determine the mutability of a resource allocation request or resource allocation: In this case the entry element is the RASD or SASD instance representing the resource allocation request or the resource allocation.

10.4.2.3.1 Preconditions

The client knows the instance path of the entry element.

10.4.2.3.2 Flow of activities

1) The client invokes the AssociatorsNames( ) intrinsic operation from the entry element through the CIM_ElementCapabilities association to obtain the set of instance paths to those CIM_AllocationCapabilities instances that represent the allocation capabilities (case (A)) or mutability (case (B)) of the entry element.
2) For each instance path obtained in step 1) the client invokes the References( ) intrinsic operation to obtain the set of instances of the CIM_SettingsDefineCapabilities association that reference each CIM_AllocationCapabilities instance from step 1).
3) For each CIM_SettingsDefineCapabilities instance obtained in step 2) the client inspects the values of the ValueRole and ValueRange properties; these values define the type of limitation imposed by the RASD instance that is referenced by the value of the PartComponent property in the CIM_SettingsDefineCapabilities association instance, as follows:
   - A default setting is designated through a value of 0 (Default) for the ValueRole property and a value of 0 (Point) for the ValueRange property. A default setting does not apply for the description of mutability.
   - A minimum setting is designated through a value of 3 (Supported) for the ValueRole property and a value of 1 (Minimums) for the ValueRange property.
   - A maximum setting is designated through a value of 3 (Supported) for the ValueRole property and a value of 2 (Maximums) for the ValueRange property.
   - An increment setting is designated through a value of 3 (Supported) for the ValueRole property and a value of 3 (Increments) for the ValueRange property.
4) For each of the CIM_SettingsDefineCapabilities association instances obtained in step 2) and inspected in step 3) the client invokes the intrinsic GetInstance( ) CIM operation, using the value of the PartComponent property as input for the InstanceName parameter. The result each time is a RASD instance where values of all non-null numeric properties describe the settings in the context established by the CIM_SettingsDefineState instance inspected in step 3).

10.4.2.3.3 Postconditions

The client knows the allocation capabilities of the system or the resource pool (case (A)), or the mutability of a resource allocation request or a resource allocation (case (B)).

10.4.2.4 Determine the default resource allocation capabilities

10.4.2.4.1 Preconditions

The client knows all of the following:

- A reference to the CIM_System instance that represents the host system.
- A selected resource type (such as for example 31 (Logical Disk) or 17 (Disk Drive))

10.4.2.4.2 Flow of activities

1) The client obtains instances of the CIM_ElementCapabilities association that reference the instance of the CIM_System class, invoking the References( ) intrinsic operation with parameter values set as follows:
   - The value of the ObjectName parameter refers to the instance of the CIM_System class.
   - The value of the ResultClass parameter is set to "CIM_ElementCapabilities"

The result of step 1) is a set of instances of the CIM_ElementCapabilities association.

2) From the result set of step 1), the client drops those instances where the value set of the Characteristics[ ] array property does not contain an element with the value 2 (Default).

The result of this step is a set of instances of the CIM_ElementCapabilities association that reference CIM_AllocationCapabilities instances that represent the default allocation capabilities of the system for a number of resource types.

3) For each of the association instances obtained in step 2), the client obtains the CIM_AllocationCapabilities instance that is referenced by the value of the Capabilities property in the respective association instance, invoking the intrinsic GetInstance( ) CIM operation with the value of the InstanceName parameter set to the value of the Capabilities property.

The result of step 3) is a set of CIM_AllocationCapabilities instances that represent the system's default allocation capabilities for a number of resource types.

4) From the result set of step 3), the client drops those instances where the value set of the ResourceType property does not match the selected resource type.

The result of this step is one RASD instance that represents the system's default allocation capabilities for the selected resource type. The client continues as in use case 10.4.2.3 step 2) in order to determine the set of RASD instances that represent the settings for the default resource allocation capabilities for the selected resource type.

10.4.2.4.3 Postconditions

The client knows the default allocation capabilities of the system for the selected resource type.

In the example CIM representation shown in Figure 36, the default allocation capabilities for the storage extent resource type of the system are represented by the CIM_AllocationCapabilities instance CAP and related RASD instances.

10.4.2.5 Determine the default resource pool

10.4.2.5.1 Preconditions

The client knows a reference to the CIM_AllocationCapabilities instance that represents the default resource allocation capabilities of the system for a selected resource type; see 10.4.2.4.

10.4.2.5.2 Flow of activities

1) The client obtains instances of the CIM_ElementCapabilities association that reference the instance of the CIM_AllocationCapabilities class, invoking the intrinsic References( ) CIM operation with parameter values set as follows:

   - The value of the ObjectName parameter refers to the CIM_AllocationCapabilities instance.
   - The value of the ResultClass parameter is set to “CIM_ElementCapabilities”.

   The result of this step is a set of instances of the CIM_ElementCapabilities association.

2) From the result set of step 1), the client drops those instances where the value set of the Characteristics[ ] array property does not contain an element with the value 2 (Default).

   The result of this step is a set of two instances of the CIM_ElementCapabilities association. One association instance references the CIM_ResourcePool instances that represent the default resource pool, and one instance references the CIM_System instance that represents the host system.

3) The client selects the instance of the CIM_ElementCapabilities association from the result of step 2) that references the CIM_ResourcePool instance by comparing the value of the ManagedElement property against the known reference to the CIM_System instance that represents the host system and dropping that association instance. The client uses the remaining association instance from the result set of step 2) to obtain the CIM_ResourcePool instance that is referenced by the value of the ManagedElement property in that association instance, invoking the intrinsic GetInstance( ) CIM operation with the value of the InstanceName parameter set to the value of the ManagedElement property.

   The result of this step is the CIM_ResourcePool instance that represents the system’s default resource pool for the selected resource type.

10.4.2.5.3 Postconditions

The client knows the default resource pool of the system for the selected resource type.

In the example CIM representation shown in Figure 36, the default storage resource pool is represented by the CIM_ResourcePool instance STOR_POOL.

10.4.2.6 Obtain the storage resource pool with the largest unreserved capacity

10.4.2.6.1 Preconditions

The client knows a reference to the CIM_System instance that represents the host system.

10.4.2.6.2 Flow of activities

1) The client resolves the CIM_HostedPool association to find the CIM_ResourcePool instances that represent resource pools hosted by the host system, invoking the intrinsic AssociatorNames( ) CIM operation with parameter values set as follows:

   - The value of the ObjectName parameter refers to the CIM_System instance that represents the host.
   - The value of the AssocClass parameter is set to "CIM_HostedPool".
   - The value of the ResultClass parameter is set to "CIM_ResourcePool".

   The result of this step is a set of CIM_ResourcePool instances that represent resource pools hosted by the host system.

2) The client selects from the result set of step 1) only those instances where the value of the ResourceType property matches 31 (Logical Disk).

   The result is a set of CIM_ResourcePool instances that represent storage resource pools hosted by the host system.

3) The client inspects the value of the Capacity and the Reserved properties in all instances selected with step 2), and each time calculates the amount of unreserved storage capacity by subtracting the value of the Reserved property from the value of the Capacity property.

4) From all pools inspected in step 3), the client selects the one that has the largest free capacity.

5) The client checks the resource pool selected in step 4) for architectural limitations as expressed by the pool's capabilities, applying use case 10.4.2.3.

10.4.2.6.3 Postconditions

The client knows the resource pool with the largest unreserved storage capacity.

In the example CIM representation shown in Figure 36, the client initially would know the CIM_System instance HOST that represents the host system. From there, the client would follow the CIM_HostedPool association to locate CIM_ResourcePool instances. Typically the association resolution would yield more than one instance, including instances that represent resource pools of other resource types; consequently the client is required to select only those instances where the value of the ResourceType property matches 31 (Logical Disk). In Figure 36, there is only one CIM_ResourcePool instance in the result set that is named STOR_POOL. From that instance, the client takes the value of the Capacity property and subtracts the value of the Reserved property (2147483648 – 268435456) byte, yielding 1879048192 blocks (or 1792 GB) as the maximum storage capacity presently available from the pool.

10.4.3 Management

This set of use cases describes how to create new virtual disks, and how to modify existing virtual disks. These management tasks are described in terms of a virtual system management service, as represented by a CIM_VirtualSystemManagementService instance.

10.4.3.1 Create virtual disk (block based)

10.4.3.1.1 Preconditions

All of the following:

- The client knows a reference to the CIM_ComputerSystem instance that represents the virtual system.
- The client knows a reference to the CIM_VirtualSystemManagementService instance that represents the virtual system management service responsible for the virtual system.
- The client has performed the use case and knows the default allocation capabilities of the system.
- The size of the new disk is 256 GB (or 268435456 blocks with a size of 1 KB (1024 bytes)).

10.4.3.1.2 Flow of activities

1) The client locally prepares a SASD instance, with properties set as follows:

   - ResourceType:31 (Logical Disk) // device type as seen by consumer
   - ResourceSubtype: // implementation dependent
   - PoolID:NULL // implies default pool

- AllocationUnits:"count" // count of blocks; if value is NULL, the effective value // is implied by pool capabilities
- VirtualQuantity:268435456 // 256 GB
- VirtualQuantityBlockSize: 1024 // may be NULL, implied by pool capabilities
- VirtualQuantityUnits: "count" // count of blocks; if value is NULL, the effective value // is implied by pool capabilities
- Reservation: NULL // may be NULL if thin provisioning is not requested; // defaults to the size expressed by the value of the // VirtualQuantity property
- Limit: NULL // defaults to maximum disk size as expressed by the // value of the VirtualQuantity property
- Address: "/dev/sda1" // optional; if not specified the implementation // assigns an address
- Values of all other properties are not set (NULL), requesting a default behavior

2) The client invokes the AddResourceSettings( ) method of the virtual system management service, with parameters set as follows:

- AffectedConfiguration: REF to the VSSD instance that represents the "defined" virtual system configuration.
- ResourceSettings: One element with the embedded SASD instance prepared in step 1)

The implementation excutes the AddResourceSettings( ) method

- It is assumed that the method returns 0, indicating successful synchronous execution.
- The initial size of the disk is 256 GB.

10.4.3.1.3 Postconditions

A new virtual disk is created for the virtual system, as requested.

10.4.3.2 Create virtual disk (file based with implicit file creation)

10.4.3.2.1 Preconditions

All of the following:

- The client knows a reference to the VSSD instance that represents the virtual system configuration to receive the new virtual disk.
- The client knows a reference to the CIM_VirtualSystemManagementService instance that represents the virtual system management service responsible for the virtual system.
- The size of the new disk is 256 GB (or 268435456 KB).
- The initial host file space to reserve is 64 GB (or 67108864 KB).
- The requested name for the file is "FILE1" (relative file path); the files does not exist initially.

10.4.3.2.2 Flow of activities

1) The client locally prepares a SASD instance, with properties set as follows:

- ResourceType: 31 (Logical Disk)

- ResourceSubtype: "DMTF:generic:scsi" // SCSI disk
- PoolID: "FILE_DISK_POOL" // implementation specific
  // dummy pool
- AllocationUnits: "count" // number of blocks
- VirtualQuantity: 268435456 // 256 GB, disk size as seen by virtual system
- Reservation: 67108864 // 64 GB, initial file size
- Limit: NULL // defaults to maximum disk size as expressed
  // by the value of the VirtualQuantity property
- HostResource[0]: "FILE://FILE1" // client chosen name and location
  // for the new file
- Address: "0.7.0.0:0" // SCSI lun 0 on target 0 via bus 0 initiator 7.
  // partition 0
- Values of all other properties are not set (NULL), requesting a default

2) The client invokes the AddResourceSettings( ) method of the virtual system management service, with parameters set as follows:

   - AffectedConfiguration: REF to the VSSD instance that represents the “defined” virtual system configuration.
   - ResourceSettings: One element with the embedded instance prepared in step 1)

   The implementation excutes the AddResourceSettings( ) method.

   - It is assumed that the method returns 0, indicating successful synchronous execution.
   - As a side effect, the new file FILE1 is created in a default directory. The initial size of the file is 64 GB, up to a limit of 256 GB. The disk size as seen by the virtual system is 256 GB from the beginning.

10.4.3.2.3 Postconditions

A new file-based virtual disk is created for the virtual system, as requested.

Figure 37 shows the situation that results after the create disk operation completed and the virtual system was activated:

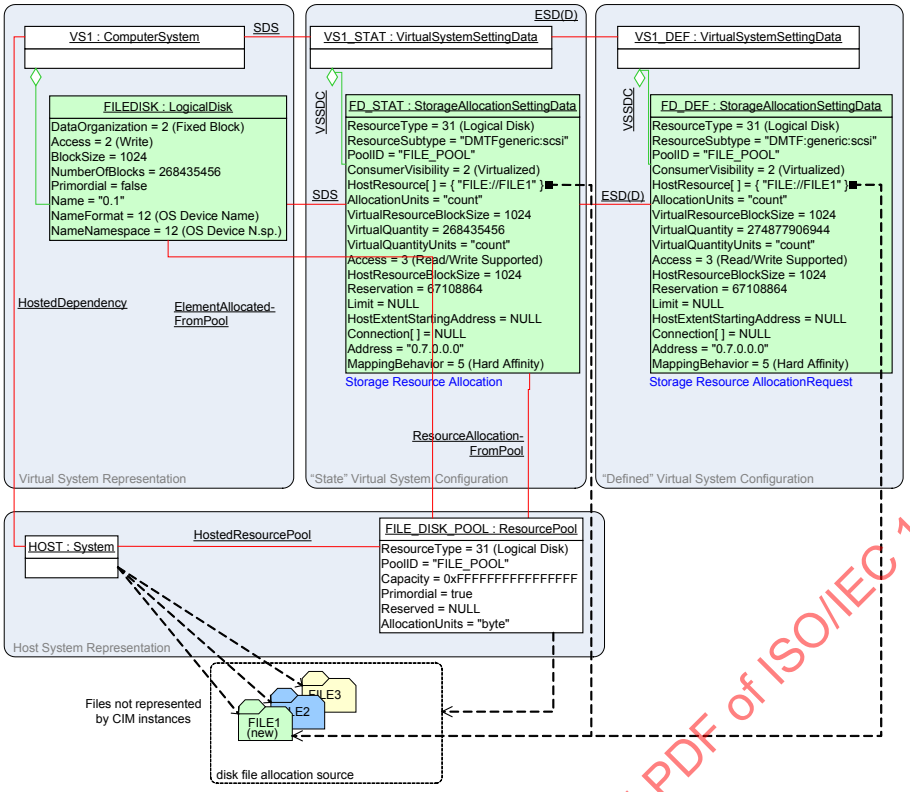

Figure 37 – Create virtual disk with implicit file creation

10.4.3.3 Create virtual disk (file based pre-existing)

10.4.3.3.1 Preconditions

All of the following:

- The client knows a reference to the VSSD instance that represents the virtual system configuration to receive the new virtual disk.
- The client knows a reference to the CIM_VirtualSystemManagementService instance that represents the virtual system management service responsible for the virtual system.
- The client knows the URI "FILE:///FILE2" of a pre-existing file that contains the data for the new disk.
- The size of the new disk is implied by the content stored in the file.

10.4.3.3.2 Flow of activities

1) The client locally prepares a SASD instance, with properties set as follows:
   - ResourceType: 31 (Logical Disk)
   - ResourceSubtype: "DMTF:generic:scsi" // Microsoft SCSI disk
   - PoolID: "FILE_DISK_POOL" // implementation specific
// dummy pool
   - AllocationUnits: "count" // number of blocks

- VirtualQuantity: NULL // disk size implied by file
- Reservation: NULL // disk reservation implied by file
- HostResource[0]: "FILE://FILE2" // URI refering to pre-existing file
- Address: "0.7.0.0.0" // SCSI bus 0 / initiator 7 / target 0 / lun 0 // partition 0
- MappingBehavior: 5 (Hard Affinity) // implies that the virtual system // requires this disk at startup
- Values of all other properties are not set (NULL), requesting a default

2) The client invokes the AddResourceSettings( ) method of the virtual system management service, with parameters set as follows:

- AffectedConfiguration: REF to the VSSD instance that represents the “Defined” virtual system configuration.
- ResourceSettings: One element with the embedded instance prepared in step 1)

The implementation executes the AddResourceSettings( ) method.

- It is assumed that the method returns 0, indicating successful synchronous execution.
- The new disk is configured into the virtual system configuration. It is based on the file provided as input. The initial disk size is 268435456 KB, as implied by the disk content stored in the file.

10.4.3.3.3 Postconditions

A new file-based virtual disk is created for the virtual system, as requested.

Figure 38 shows the situation that results after the create disk operation completed and the virtual system was activated:

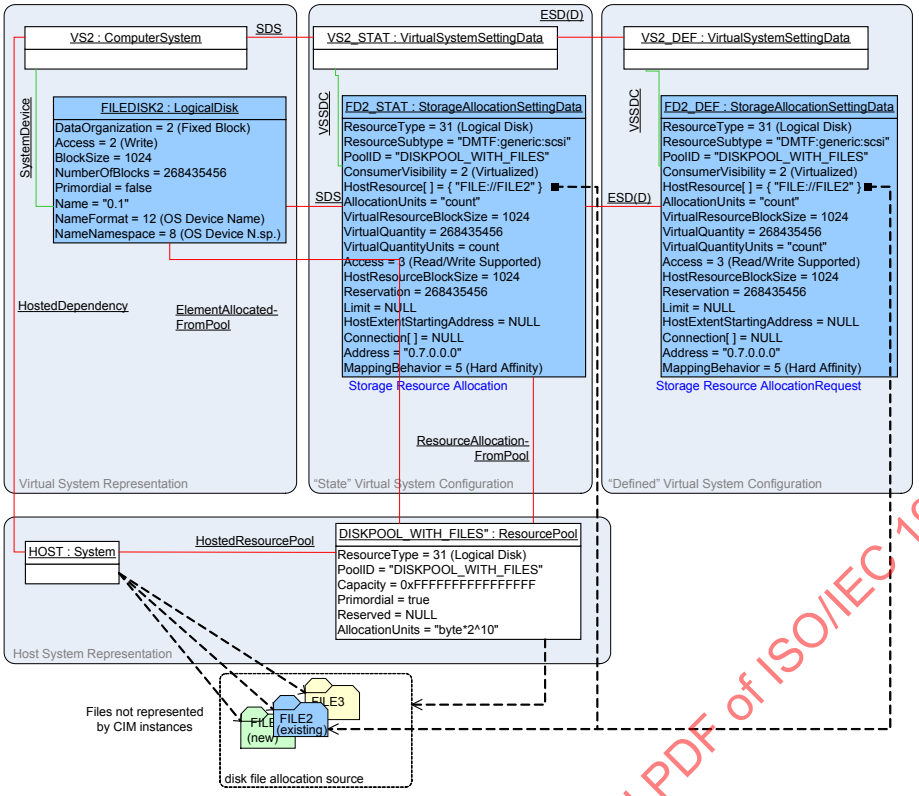

Figure 38 – Create virtual disk with pre-existing file

10.4.3.4 Create virtual disk (block based passed-through)

10.4.3.4.1 Preconditions

All of the following:

- The client knows a reference to the VSSD instance that represents the virtual system configuration to receive the new dedicated virtual disk.
- The client knows a reference to the CIM_VirtualSystemManagementService instance that represents the virtual system management service responsible for the virtual system.
- The size of the new disk is 256 GB (or 268435456 kbyte).

10.4.3.4.2 Flow of activities

1) The client locally prepares an instance of the CIM_ResourceAllocationSettingData class, with properties set as follows:

- ResourceType: 31 (Logical Disk)
- ResourceSubtype: "DMTF:xen:vbd"
- PoolID: "PT_POOL" // Example; refers to a pool withdisks
// that are passed-through
- AllocationUnits: "count" // number of blocks

- VirtualQuantity: 268435456 // 256 GB; needed if not explicit host resource // requested
- VirtualResourceBlockSize: 1024 // blocksize as seen by consumer
- VirtualQuantityUnits: "count" // number of blocks
- Reservation: 274877906944 // needed if no explicit // host resource requested
- HostResource[0]: "URI(HDISK1)" // optional; may refer to a // specific disk in the pool; if not // specified, the implementation // selects a disk out of the pool
- Address: "/dev/sda1" // optional; if not specified the // implementation assigns an // address
- Values of all other properties are not set (or are NULL), requesting a default

2) The client invokes the AddResourceSettings( ) method of the virtual system management service, with parameters set as follows:

- AffectedConfiguration: REF to CIM_VirtualSystemSettingData instance
- ResourceSettings: One element with the embedded instance prepared in step 1)

It is assumed that the method returns 0, indicating successful synchronous execution.

10.4.3.4.3 Postconditions

A new passed-through virtual disk is created for the virtual system, as requested.

Figure 39 shows the situation that results after the AddResourceSettings( ) operation completes.

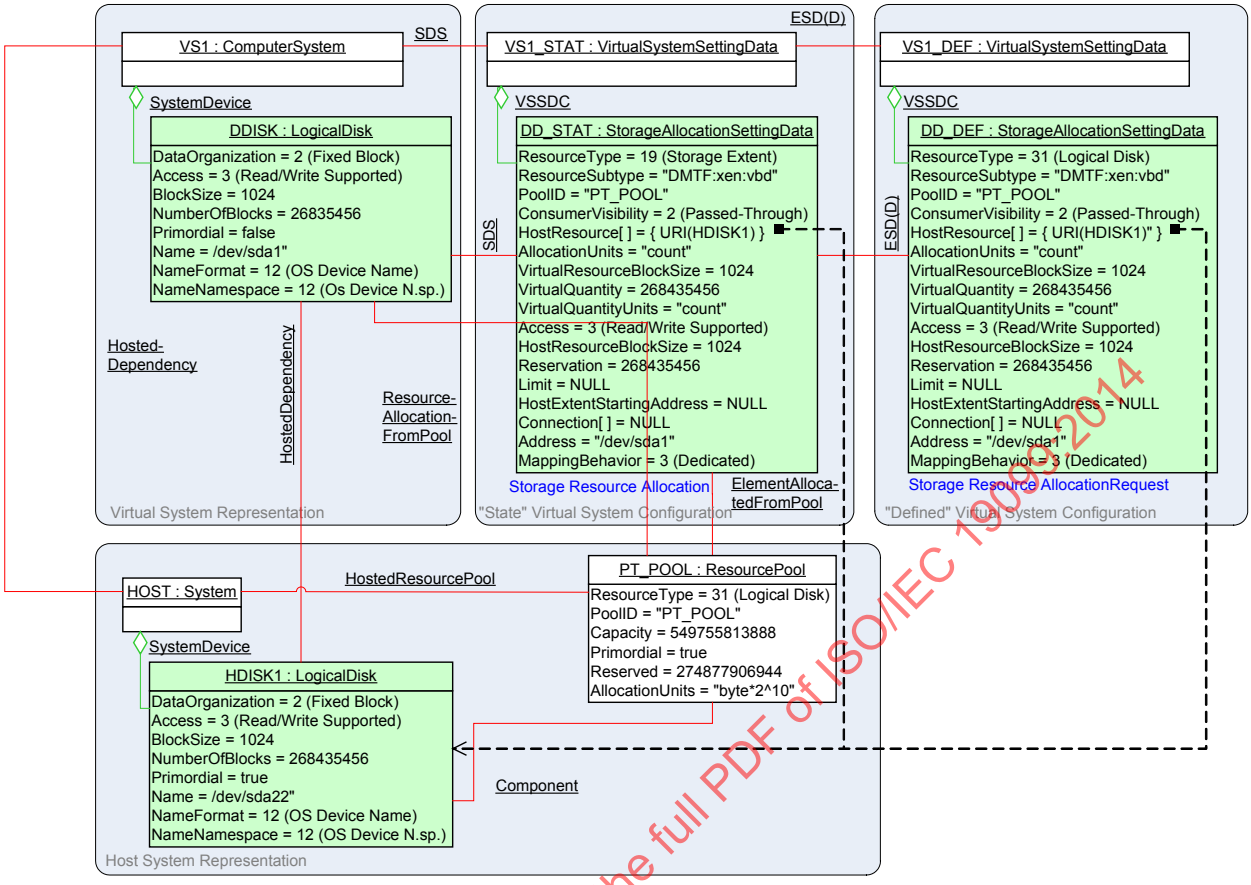

Figure 39 – Create dedicated virtual disk

In this example the resource pool represented by PT_POOL aggregates a set of host disks that were set aside for the purpose of being passed-through to virtual systems. Adding a resource from that pool is actually a selection based upon client requirements.

10.4.3.5 Create virtual disk (file based delta)

10.4.3.5.1 Preconditions

All of the following:

- The situation that was the result of the use case described in 10.4.3.2.
- The size of the virtual remains disk is 256 GB (or 268435456 KB).
- The initial size of the new delta disk is 16 GB (or 16777216 KB).
- The name of the file is "FILE9" (relative file path); the file does not exist.
- The file is only allocated as the virtual system is activated (instantiated).
- The file is deallocated as the virtual system is deactivated.

10.4.3.5.2 Flow of activities

1) The client locally prepares a SASD instance with properties set as follows:
   - ResourceType: 31 (Logical Disk)

- ResourceSubtype: "DMTF:generic:scsi" // SCSI disk
- PoolID: "FILE_DISK_POOL" // implementation specific
// dummy pool
- AllocationUnits: "count" // count of blocks
- VirtualQuantity: 268435456 // 256 GB
- Reservation: 16777216 // 16 GB
- Limit: 67108864 // 64 GB
- AutomaticAllocation: true // fresh extent allocated
// on every allocation
- AutomaticDeallocation: true // extent dropped at deallocation time
- HostResource[0]: "FILE://FILE9" // optional; if NULL the
// implementation decides
- Parent: URI (FD_DEF) // formatted as specified in DSP0207
- Address: "0.7.0.0.0" // SCSI bus 0 / initiator 7 / target 0 / lun 0
// partition 0
- Values of all other properties are not set (or are NULL), requesting a default

2) The client invokes the AddResourceSettings( ) method of the virtual system management service, with parameters set as follows:

- AffectedConfiguration: REF to VSSD instance // target config
- ResourceSettings: One element with the embedded instance prepared in step 1)

It is assumed that the method that the method returns 0, indicating successful synchronous execution.

10.4.3.5.3 Postconditions

A new delta file-based virtual disk is created for the virtual system, as requested.

Figure 40 shows the situation that results after the create delta disk operation completes.

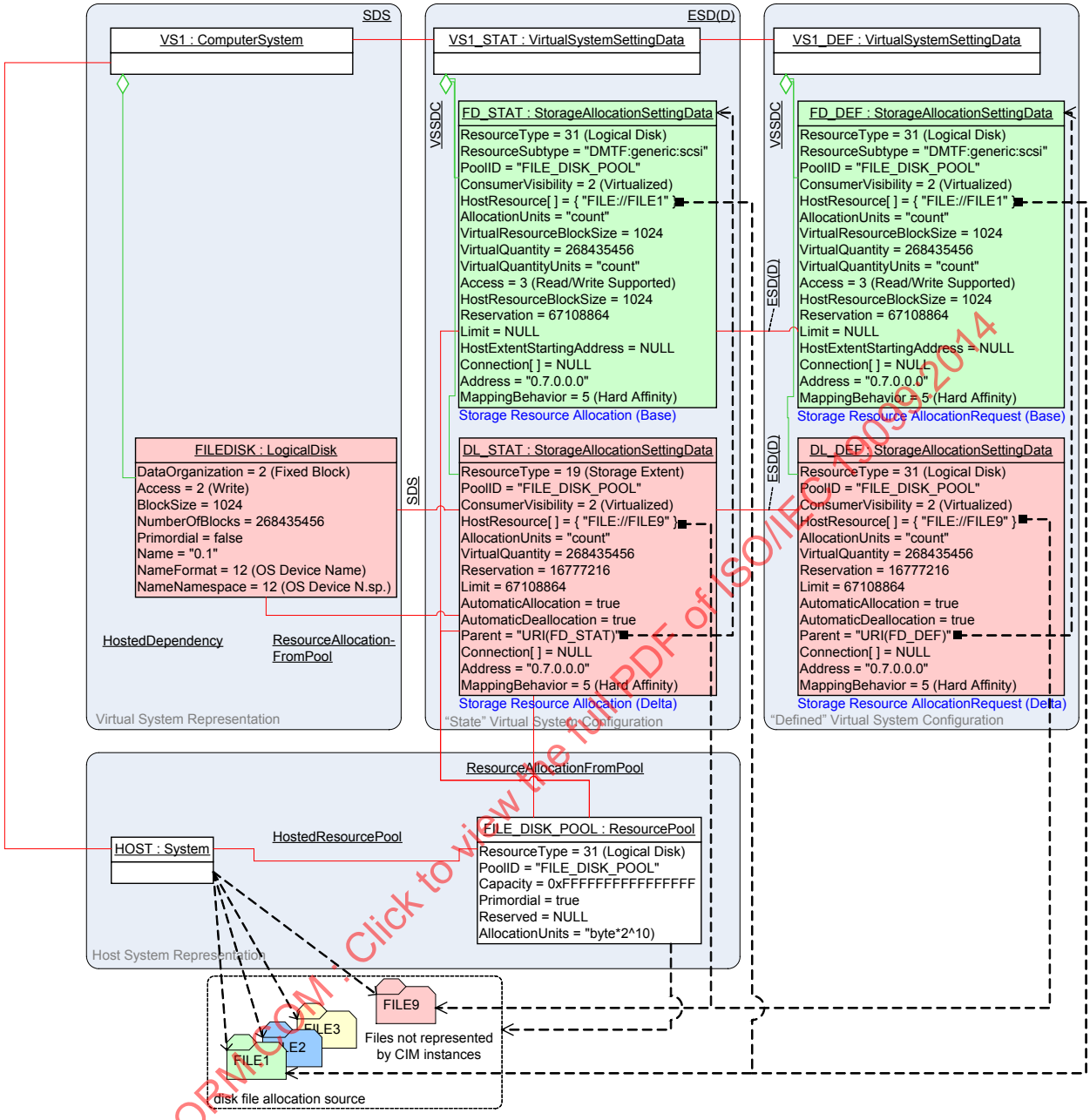

Figure 40 – Create virtual delta disk and file

Note that the instances FD_STAT, DL_STAT and FILEDISK are present only while the virtual system is instantiated and the virtual disk is allocated. Note that there is only one disk FILEDISK in the virtual system representation that is allocated based on both FD_STAT and DL_STAT. There is no separate instance of CIM_LogicalDisk representing each allocation separately as there is only one virtual disk presented to the virtual system.

Note that the file FILE9 containing the delta disk is automatically allocated during virtual disk allocation because the value of the AutomaticAllocation property is true; the file is automatically deallocated during virtual disk deallocation because the value of the AutomaticDeallocation property is true. As a consequence the virtual system at startup time receives a virtual disk that is initially based on FILE1; as

the virtual system writes onto the disk the delta is maintained in FILE9. The size of FILE9 is driven by the values of the Reservation and the Limit properties in DL_STAT: The initial file size is 16 GB, up to a limit of 64 GB. As a result the virtual system sees a disk with a size of 256 GB (as indicated by the value of the VirtualQuantity property). That disk is initially based on the read-only file-based extent as allocated by FD_STAT. On top of the read-only extent is a temporary delta read-write extent as allocated by DL_STAT that enables overwriting data up to an amount of 64 GB; the delta extent is discarded when the virtual disk is deallocated, such that the next allocation starts with the initial read-only content again.

10.5 CIM Elements

Table 149 lists CIM elements that are defined or specialized for the profile described in this clause. Each CIM element shall be implemented as described in Table 149. The CIM Schema descriptions for any referenced element and its sub-elements apply.

Subclauses 10.2 ("Implementation") and 10.3 ("Methods") may impose additional requirements on these elements; in particular, subclause 10.2 ("Implementation") may impose requirements for CIM instances.

Table 149 – CIM Elements: Storage Resource Virtualization Profile

Element	Requirement	Description
Classes		
CIM_AffectedJobElement	Optional	See the Resource Allocation Profile described in clause 5.
CIM_AllocationCapabilities for capabilities	Mandatory	See the Allocation Capabilities Profile described in clause 7.
CIM_AllocationCapabilities for mutability	Optional	See the Allocation Capabilities Profile described in clause 7.
CIM_Component for resource pool	Conditional	See 10.5.1.
CIM_ConcreteJob	Optional	See the Resource Allocation Profile described in clause 5.
CIM_DiskDrive for host disk drives	Conditional	See 10.5.2.
CIM_DiskDrive for virtual disk drives	Conditional	See 10.5.3.
CIM_ElementAllocatedFromPool for allocated virtual resources	Mandatory	See 0.
CIM_ElementAllocatedFromPool for resource pool hierarchies	Conditional	See 10.5.5.
CIM_ElementCapabilities for capabilities	Mandatory	See the Allocation Capabilities Profile described in clause 7.
CIM_ElementCapabilities for mutability	Conditional	See the Allocation Capabilities Profile described in clause 7.
CIM_ElementCapabilities for resource pool	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_ElementSettingData for resource allocation request	Mandatory	See 0.
CIM_ElementSettingData for resource pool	Mandatory	See 10.5.7.
CIM_HostedDependency	Optional	See 0.

Element	Requirement	Description
CIM_HostedResourcePool	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_HostedService	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_LogicalDisk for virtual disk	Conditional	See 10.5.9.
CIM_ReferencedProfile	Mandatory	See 10.5.10.
CIM_RegisteredProfile	Mandatory	See 10.5.11.
CIM_ResourceAllocationFromPool	Optional	See the Resource Allocation Profile described in clause 5.
CIM_ResourceAllocationSettingData for disk drive allocation information	Conditional	See 10.5.12.
CIM_ResourcePool	Mandatory	See 10.5.13.
CIM_ResourcePoolConfigurationCapabilities	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_ResourcePoolConfigurationService	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_SettingsDefineCapabilities	Mandatory	See the Allocation Capabilities Profile described in clause 7.
CIM_SettingsDefineState	Mandatory	See 10.5.14.
CIM_ServiceAffectsElement	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_StorageAllocationSettingData for storage extent allocation information	Conditional	See 10.5.15.
CIM_StorageVolume for host storage volume	Conditional	See 10.5.16.
CIM_StorageExtent for virtual disk	Conditional	See 10.5.17.
CIM_SystemDevice for host storage volumes	Conditional	See 10.5.18.
CIM_SystemDevice for virtual resources	Mandatory	See 10.5.19.
Indications		
None defined		

10.5.1 CIM_Component for resource pool

The implementation of the CIM_Component association for the representation of the aggregation of host resources into resource pools is conditional.

Condition: The resource aggregation feature (see 10.2.5) is implemented.

The CIM_Component association is abstract; therefore it cannot be directly implemented. For this reason the provisions in this subclause shall be applied to implementations of subclasses of the CIM_Component association. However, note that clients may directly resolve abstract associations without knowledge of the concrete subclass that is implemented.

Table 150 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 150 – Association: CIM_Component for resource pool

Elements	Requirement	Notes
GroupComponent	Mandatory	**Key:** Value shall reference the CIM_ResourcePool instance that represents the resource pool. **Cardinality:** 1
PartComponent	Mandatory	**Key:** Value shall reference the CIM_ManagedElement instance that represents a component of the resource pool. **Cardinality:** *

10.5.2 CIM_DiskDrive for host disk drives

The implementation of the CIM_DiskDrive class for the representation of host disk drives is conditional.

Condition: The resource aggregation feature is implemented for disk drive resource pools; see 10.2.5.

Table 151 lists the requirements for elements of this class.

Table 151 – Class: CIM_DiskDrive (Host)

Elements	Requirement	Notes
DefaultBlockSize	Mandatory	See CIM schema description

10.5.3 CIM_DiskDrive for virtual disk drives

The implementation of the CIM_DiskDrive class for the representation of virtual disk drives is conditional.

Condition: The profile described in this clause is implemented for the allocation of disk drives; see 10.2.2.

Table 152 lists the requirements for elements of this class.

Table 152 – Class: CIM_DiskDrive (Virtual System)

Elements	Requirement	Notes
EnabledState	Mandatory	Value shall match { 2 | 3 } ("Enabled" | "Disabled").
RequestedState	Optional	Value shall match { 2 | 3 } ("Enabled" | "Disabled").
DefaultBlockSize	Mandatory	See CIM schema description

10.5.4 CIM_ElementAllocatedFromPool for allocated virtual resources

Table 153 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 153 – Association: CIM_ElementSettingData

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Value shall reference the CIM_ResourcePool instance that represents the resource pool. **Cardinality:** 1
Dependent	Mandatory	**Key:** Value shall reference the instance of a CIM_LogicalDevice subclass that represents the allocated device. **Cardinality:** *

10.5.5 CIM_ElementAllocatedFromPool for resource pool hierarchies

The implementation of the CIM_ElementAllocatedFromPool association for the representation of resource pool hierarchies is conditional.

Condition: The resource pool management feature (see 10.2.7) is implemented.

Table 154 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 154 – Association: CIM_ElementSettingData

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Value shall reference the CIM_ResourcePool instance that represents the parent resource pool. **Cardinality:** 1
Dependent	Mandatory	**Key:** Value shall reference the CIM_ResourcePool instance that represents the child resource pool. **Cardinality:** *

10.5.6 CIM_ElementSettingData for resource allocation request

Table 155 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 155 – Association: CIM_ElementSettingData

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Value shall reference the RASD instance that represents the resource allocation. **Cardinality:** 1
SettingData	Mandatory	**Key:** Value shall reference the RASD instance that represents corresponding the resource allocation request. **Cardinality:** 1
IsDefault	Mandatory	Value shall be 1 (Is Default).

10.5.7 CIM_ElementSettingData for resource pool

Table 156 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 156 – Association: CIM_ElementSettingData

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Value shall reference the CIM_ResourcePool instance that represents a child resource pool. **Cardinality:** 1
SettingData	Mandatory	**Key:** Value shall reference the RASD instance that represents corresponding the resource allocation request. **Cardinality:** 1

10.5.8 CIM_HostedDependency

The implementation of the CIM_HostedDependency association is optional.

Table 157 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 157 – Association: CIM_HostedDependency

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Value shall reference the instance of the CIM_LogicalDevice class that represents a dedicated host device. **Cardinality:** 0..1
Dependent	Mandatory	**Key:** Value shall reference the instance of the CIM_LogicalDevice class that represents a virtual device. **Cardinality:** 0..1

10.5.9 CIM_LogicalDisk for virtual disk

The implementation of the CIM_LogicalDisk class for the representation of virtual disks is conditional.

Condition: The profile described in this clause is implemented for the allocation of storage extents; see 10.2.2.

Table 158 lists the requirements for elements of this class in addition to those specified for the implementation of the CIM_StorageExtent class for the representation of virtual disks; see 10.5.17.

Table 158 – Class: CIM_LogicalDisk (Virtual System)

Elements	Requirement	Notes
Name	Mandatory	See 10.2.9.2.3.
NameFormat	Mandatory	See 10.2.9.2.4.
NameNamespace	Mandatory	See 10.2.9.2.5.

10.5.10 CIM_ReferencedProfile

Table 159 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in DSP1033.

Table 159 – Association: CIM_ReferencedProfile

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Value shall reference the CIM_RegisteredProfile instance that represents an implementation of the profile described in this clause. **Cardinality:** 0..1
Dependent	Mandatory	**Key:** Value shall reference the CIM_RegisteredProfile instance that represents an implementation of the scoping profile. **Cardinality:** 0..1

10.5.11 CIM_RegisteredProfile

Table 160 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM schema and in DSP1033.

Table 160 – Class: CIM_RegisteredProfile

Elements	Requirement	Notes
RegisteredOrganization	Mandatory	Value shall be 2 (DMTF).
RegisteredName	Mandatory	Value shall be "Storage Resource Virtualization".
RegisteredVersion	Mandatory	Value shall be "1.0.0".

10.5.12 CIM_ResourceAllocationSettingData for disk drive allocation information

The implementation of the CIM_ResourceAllocationSettingData class for the representation of disk drive allocation information is conditional.

Condition: The profile described in this clause is implemented for the allocation of disk drives; see 10.2.2.

Table 161 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 161 – Class: CIM_ResourceAllocationSettingData

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**.
ResourceType	Mandatory	See 10.2.8.3.1.
OtherResourceType	Mandatory	Value shall be NULL.
ResourceSubType	Optional	See 10.2.8.3.2.
PoolID	Mandatory	See 10.2.8.3.3.
ConsumerVisibility	Optional	See 10.2.8.3.4.
HostResource[ ]	Conditional	See 10.2.8.3.5.

Elements	Requirement	Notes
AllocationUnits	Mandatory	See 10.2.8.3.6.
VirtualQuantity	Mandatory	See 10.2.8.3.7.
VirtualQuantityUnits	Mandatory	EXPERIMENTAL; See 10.2.8.3.8.
Reservation	Optional	See 10.2.8.3.9.
Limit	Optional	See 10.2.8.3.10.
Weight	Optional	See 10.2.8.3.11.
AutomaticAllocation	Optional	See the Resource Allocation Profile described in clause 5.
AutomaticDeallocation	Optional	See the Resource Allocation Profile described in clause 5.
Parent	Optional	See 10.2.8.3.12.
Connection[ ]	Optional	See 10.2.8.3.13.
Address	Optional	See 10.2.8.3.14.
MappingBehavior	Optional	See 10.2.8.3.15.

10.5.13 CIM_ResourcePool

Table 162 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 162 – Class: CIM_ResourcePool

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
PoolID	Mandatory	See 10.2.4.4.
Primordial	Mandatory	See 10.2.4.5.
Capacity	Conditional	See 10.2.4.7.
Reserved	Optional	See 10.2.4.6.
ResourceType	Mandatory	See 10.2.4.2.
OtherResourceType	Mandatory	Value shall be NULL.
ResourceSubType	Optional	See 10.2.4.3.
AllocationUnits	Mandatory	See 10.2.4.8.
MaxConsumableResource	Optional	See 10.2.4.9.
CurrentlyConsumedResource	Optional	See 10.2.4.10.
ConsumedResourceUnit	Optional	See 10.2.4.11.

10.5.14 CIM_SettingsDefineState

Table 163 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 163 – Association: CIM_SettingsDefineState

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Value shall reference the CIM_ManagedSystemElement instance representing the allocated virtual resource. **Cardinality:** 0..1
SettingData	Mandatory	**Key:** Value shall reference the CIM_ResourceAllocationSettingData instance representing the resource allocation. **Cardinality:** 0..1

10.5.15 CIM_StorageAllocationSettingData for storage allocation information

The implementation of the CIM_StorageAllocationSettingData class for the representation of storage allocation information is conditional.

Condition: The profile described in this clause is implemented for the allocation of storage extents; see 10.2.2.

Table 164 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 164 – Class: CIM_StorageAllocationSettingData

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**.
ResourceType	Mandatory	See 10.2.8.4.1.
OtherResourceType	Mandatory	Value shall be NULL.
ResourceSubType	Optional	See 10.2.8.3.2.
PoolID	Mandatory	See 10.2.8.3.3.
ConsumerVisibility	Optional	See 10.2.8.3.4.
HostResource[]	Conditional	See 10.2.8.4.5.
AllocationUnits	Mandatory	See 10.2.8.4.6.
VirtualQuantity	Optional for Q_SASD Mandatory otherwise	See 10.2.8.4.7.
VirtualQuantityUnit	Mandatory	See 10.2.8.4.8.
Reservation	Optional	See 10.2.8.4.9.
Limit	Optional	See 10.2.8.4.10.
Weight	Optional	See 10.2.8.3.11.
AutomaticAllocation	Optional	See the Resource Allocation Profile described in clause 5.

Elements	Requirement	Notes
AutomaticDeallocation	Optional	See the Resource Allocation Profile described in clause 5.
Parent	Optional	See 10.2.8.3.12.
Connection[ ]	Optional	See 10.2.8.3.13.
Address	Optional	See 10.2.8.3.14.
MappingBehavior	Optional	See 10.2.8.3.15.
VirtualResourceBlockSize	Mandatory	See 10.2.8.4.16.
Access	Optional	See 10.2.8.4.17.
HostResourceBlockSize	Mandatory	See 10.2.8.4.18.
HostExtentStartingAddress	Optional	See 10.2.8.4.19.
HostExtentName	Optional	See 10.2.8.4.20.
HostExtentNameFormat	Conditional	See 10.2.8.4.21.
OtherHostExtentNameFormat	Conditional	See 10.2.8.4.22.
HostExtentNameNamespace	Conditional	See 10.2.8.4.23.
OtherHostExtentNameNamespace	Conditional	See 10.2.8.4.24.

10.5.16 CIM_StorageVolume for host storage volume

The implementation of the CIM_StorageVolume class for the representation of host storage volumes is conditional.

Condition: The storage resource aggregation feature is implemented; see 10.2.5.

Table 165 lists the requirements for elements of this class.

Table 165 – Class: CIM_StorageVolume for host storage volume

Elements	Requirement	Notes
Access	Mandatory	See CIM Schema description.
BlockSize	Mandatory	See CIM Schema description.
NumberOfBlocks	Mandatory	See CIM Schema description.
Name	Mandatory	See CIM Schema description.
NameFormat	Mandatory	See CIM Schema description.
NameNamespace	Mandatory	See CIM Schema description.

10.5.17 CIM_StorageExtent for virtual storage extent

See 10.2.9 for detailed implementation requirements for this class if it is used for the representation of virtual disks.

Table 166 lists the requirements for elements of this class.

Table 166 – Class: CIM_StorageExtent for virtual disks

Elements	Requirement	Notes
BlockSize	Mandatory	See 10.2.9.2.1.
NumberOfBlocks	Mandatory	Value shall reflect the number of blocks available to the virtual system.
Name	Mandatory	Value may reflect the name of the virtual disk.
NameFormat	Mandatory	See CIM schema description.
NameNamespace	Mandatory	See CIM schema description.

10.5.18 CIM_SystemDevice for host storage volumes

The implementation of the CIM_SystemDevice association for host storage volumes is conditional.

Condition: The storage resource aggregation feature is implemented; see 10.2.5.

Table 167 lists the requirements for elements of this association.

Table 167 – Association: CIM_SystemDevice for host storage volumes

Elements	Requirement	Notes
GroupComponent	Mandatory	**Key:** Value shall reference an instance of the CIM_System class. **Cardinality:** 1
PartComponent	Mandatory	**Key:** Value shall reference the instance of the CIM_StorageVolume class. **Cardinality:** *

10.5.19 CIM_SystemDevice for virtual resources

Table 168 lists the requirements for elements of this association.

Table 168 – Association: CIM_SystemDevice for virtual resources

Elements	Requirement	Notes
GroupComponent	Mandatory	**Key:** Value shall reference an instance of the CIM_ComputerSystem class representing the virtual system. **Cardinality:** 1
PartComponent	Mandatory	**Key:** Value shall reference the instance of the CIM_LogicalDisk, CIM_StorageVolume, CIM_StorageExtent or CIM_DiskDrive class representing the virtual resource. **Cardinality:** *

11 Ethernet Port Resource Virtualization Profile

Profile Name: Ethernet Port Resource Virtualization

Profile Version: 1.0.0

Organization: DMTF

CIM Schema Version: 2.26

Central Class: CIM_ResourcePool

Scoping Class: CIM_System

The profile described in this clause is a component profile that defines the minimum object model needed to provide for the CIM representation and management of the virtualization of Ethernet ports and connections.

Table 169 lists DMTF management profiles on which the Ethernet Port Resource Virtualization Profile depends.

Table 169 – Related profiles for the Ethernet Port Resource Virtualization Profile

Profile Name	Organization	Version	Requirement	Description
Resource Allocation	DMTF	1.1	Specializes	The abstract profile that describes the virtualization of resources See clause 5.
Allocation Capabilities	DMTF	1.0	Specializes	The abstract profile that describes capabilities for resource allocation See clause 7.
Profile Registration	DMTF	1.0	Mandatory	The profile that specifies registered profiles See DSP1033.
Ethernet Port	DMTF	1.0	Optional	The profile that specifies the management of Ethernet Ports See DSP1014.

11.1 Description

This subclause contains informative text only It introduces the management domain addressed by the profile described in this clause and outlines the central modeling elements established for representation and control of the management domain.

11.1.1 General

In computer virtualization systems, virtual computer systems are composed of component virtual resources. The profile described in this clause specializes the resource virtualization pattern as defined in the Resource Allocation Profile described in clause 5. and the allocation capabilities pattern as defined in the Allocation Capabilities Profile described in clause 7 for the representation and management of the following types of resources:

- Ethernet adapters, designated by resource type value 10 (Ethernet Adapter). Ethernet adapters are allocated to a virtual computer system.
- Ethernet switch ports, designated by resource type value 30 (Ethernet Switch Port). Ethernet switch ports are allocated to virtual Ethernet switches.
- Ethernet connections, designated by resource type value 33 (Ethernet Connection). Ethernet connections represent the connection (association CIM_ActiveConnection) between two CIM_LANEndpoint instances that are associated to the instances of CIM_EthernetPort representing either an Ethernet adapter or an Ethernet switch port.

The profile described in this clause references additional or specialized CIM elements and extends constraints beyond those defined in the abstract profiles.

11.1.2 Ethernet port resource virtualization class schema

Figure 41 shows the class schema of the profile described in this clause. It outlines the elements that are referenced and in some cases further constrained by the profile described in this clause, as well as the dependency relationships between elements of the profile described in this clause and other profiles. For simplicity in diagrams, the *CIM_* prefix has been removed from class and association names. Inheritance relationships are shown only to the extent required in the context of the profile described in this clause.

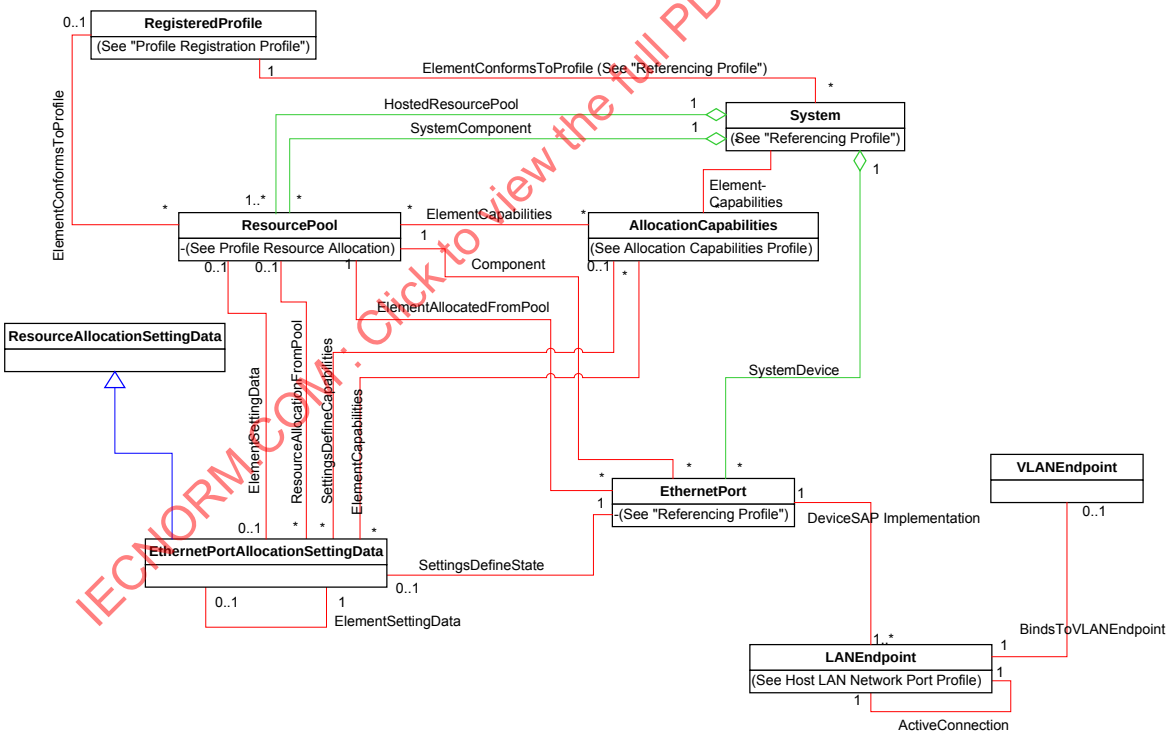

Figure 41 – Ethernet Port Resource Virtualization: Profile class diagram

The profile described in this clause specializes the Resource Allocation Profile described in clause 5 and the Allocation Capabilities Profile described in clause 7 by defining more specific adaptations for the following classes and associations:

- The CIM_ResourcePool class models the resource pools for Ethernet resources. The resource pool is used to allocate the resources required to instantiate virtual Ethernet adapters and Ethernet switch ports that are modeled by the CIM_EthernetPort class.
- The CIM_ResourcePool class also models the resources required to describe a connection between the LAN endpoints of an Ethernet adapter and an Ethernet switch port.
- The CIM_Component association models the relationship between resource pools (with a type of either Ethernet adapters or Ethernet switch port) and host Ethernet ports as components of the resource pools.
- The CIM_ElementAllocatedFromPool association models the relationship between resource pools and the virtual Ethernet ports allocated from those pools.
- The CIM_ResourceAllocatedFromPool association models the relationship between a resource pool and the resource allocations provided by the resource pool.
- The CIM_HostedResourcePool association models the hosting dependency between a resource pool and its host system.
- The CIM_EthernetPort class models the following aspects of both an Ethernet adapter and an Ethernet switch port:
  - CIM_EthernetPort as a device in the scope of a system (computer system or virtual Ethernet switch), as modeled by the CIM_SystemDevice association
  - CIM_EthernetPort as a result of an Ethernet adapter or Ethernet switch port resource allocation from a resource pool, as modeled by the CIM_ElementAllocatedFromPool association
  - CIM_EthernetPort as a component within Ethernet adapter or Ethernet switch port resource pools, as modeled by the CIM_Component association
- The CIM_EthernetPortAllocationSettingData class is a subclass of the CIM_ResourceAllocationSettingData class and models
  - Ethernet adapter resource allocations or allocation requests
  - Ethernet switch port resource allocations or allocation requests
  - Ethernet connection allocations or allocation requests. Ethernet connection resource allocations or allocation requests represent an allocation request for the connection between a pair of CIM_LANEndpoint instances or a current allocation of the described connection.
- The CIM_ElementSettingData association between the classes CIM_EthernetPort and CIM_EthernetPortAllocationSettingData models the relationship between an Ethernet adapter represented by the class CIM_EthernetPort and an Ethernet connection allocation represented by the class CIM_EthernetPortAllocationSettingData. This use of the association is in compliance with a simple allocation as described in the Resource Allocation Profile described in clause 5.
- The CIM_AllocationCapabilities class and the CIM_ElementCapabilities association model:
  - the resource allocation capabilities of the host system and/or a resource pool for resource types 10 (Ethernet Adapter) or 30 (Ethernet Switch Port)
  - the mutability of existing allocations for resource types 10 (Ethernet Adapter) or 30 (Ethernet Switch Port)

- the allocation capabilities of the host systems and/or resource pools for resource type 33 (Ethernet Connection)
- the mutability of existing allocations for resource type 33 (Ethernet Connection)

In general, any mention of a class in this document means the class itself or its subclasses. For example, a statement such as “an instance of the CIM_LogicalDevice class” implies an instance of the CIM_LogicalDevice class or a subclass of the CIM_LogicalDevice class.

11.1.3 Resource pools

The profile described in this clause applies the concept of resource pools defined in the Resource Allocation Profile described in clause 5 to resource types 10 (Ethernet Adapter), 30 (Ethernet Switch Port), and 33 (Ethernet Connection).

The profile described in this clause uses the Ethernet port resource pool as the focal point for Ethernet adapter and Ethernet switch port allocations. These are respectively allocated to virtual computer systems as defined in the Virtual System Profile described in clause 12 and Ethernet switches as defined in the Virtual Ethernet Switch Profile described in clause 14.

The profile described in this clause uses Ethernet connection resource pools are the focal point for the allocation of Ethernet connections. These are allocated to establish the connection between the LAN Endpoints associated to an Ethernet adapter and that implemented by an Ethernet switch port.

11.1.3.1 General

The profile described in this clause applies the concept of resource pools defined in 5.1.1.2 to the following resource types:

- Resource type 10 (Ethernet Adapter) designates Ethernet adapter resource pools that represent resources for the allocation of Ethernet adapters for the use by virtual systems; allocated Ethernet adapters are represented by CIM_EthernetPort instances.
- Resource type 30 (Ethernet Switch Port) designates Ethernet switch port resource pools that represent resources for the allocation of Ethernet switch ports for use by virtual Ethernet switches; allocated Ethernet switch ports are represented by CIM_EthernetPort instances.
- Resource type 33 (Ethernet Connection) designates Ethernet connection resource pools that represent resources for the allocation of connections between an Ethernet adapter that is a resource of a virtual system and an Ethernet switch port that is a resource of a virtual Ethernet switch.

The resource type of a resource pool governs the resource types that are allocated from the resource pool. The type of host resources that are aggregated by a resource pool may differ from the resource type of the pool. For example, a resource pool with a resource type of 10 (Ethernet Adapter) supports the allocation of virtual Ethernet adapters. However, the resources that are aggregated by that resource pool may be of a different type; for example, the resource pool might simply represent connectivity to an external network.

11.1.3.2 Representation of host resources

Resource pools for Ethernet adapters or Ethernet switch ports represent host resources that enable the allocation of respective virtual devices, namely virtual Ethernet adapters or virtual Ethernet switch ports; resource pools for Ethernet connections represent host resources that enable the allocation of virtual Ethernet connections. However, the explicit representation of the host resources aggregated by a resource pool is optional. In some cases, implementations may explicitly represent the host resources, such as host Ethernet adapters or host Ethernet switch ports. In other cases, implementations may choose not to explicitly represent the host resources aggregated by a resource pool. For example, an implementation for the representation and management of virtual Ethernet connections is not required to

explicitly model the host resources that support the virtual Ethernet connections; instead, in this case, the resource pool is the sole model element that represents the Ethernet connection capacity assigned for the support of (allocated) virtual Ethernet connections and the capacity that is still available for the allocation of new Ethernet connections.

11.1.4 Resource allocation

This subclause describes how the profile described in this clause models resource allocations and resource allocation requests for Ethernet resources.

11.1.4.1 General

The profile described in this clause specializes the concept of *virtual resource allocation* defined in 5.1.3 to resource types 10 (Ethernet Adapter) and 30 (Ethernet Switch Port), both modeled by the CIM_EthernetPort class.

The profile described in this clause specializes the concept of *simple resource allocation* defined in 5.1.2 to resource type 33 (Ethernet Connection). Simple resource allocation implies that the result of the allocation is not represented by a CIM_LogicalDevice instance.

11.1.4.2 Ethernet resource allocation for virtual ethernet switches

Figure 42 shows an example of the allocation of an Ethernet switch port to a virtual switch. The upper part of Figure 42 shows a static allocation request of a virtual Ethernet switch port to a virtual Ethernet switch, applying the concept of virtual resource allocation as specified in 5.2.2. The lower part of Figure 42 shows the virtual switch with the allocated Ethernet switch port.

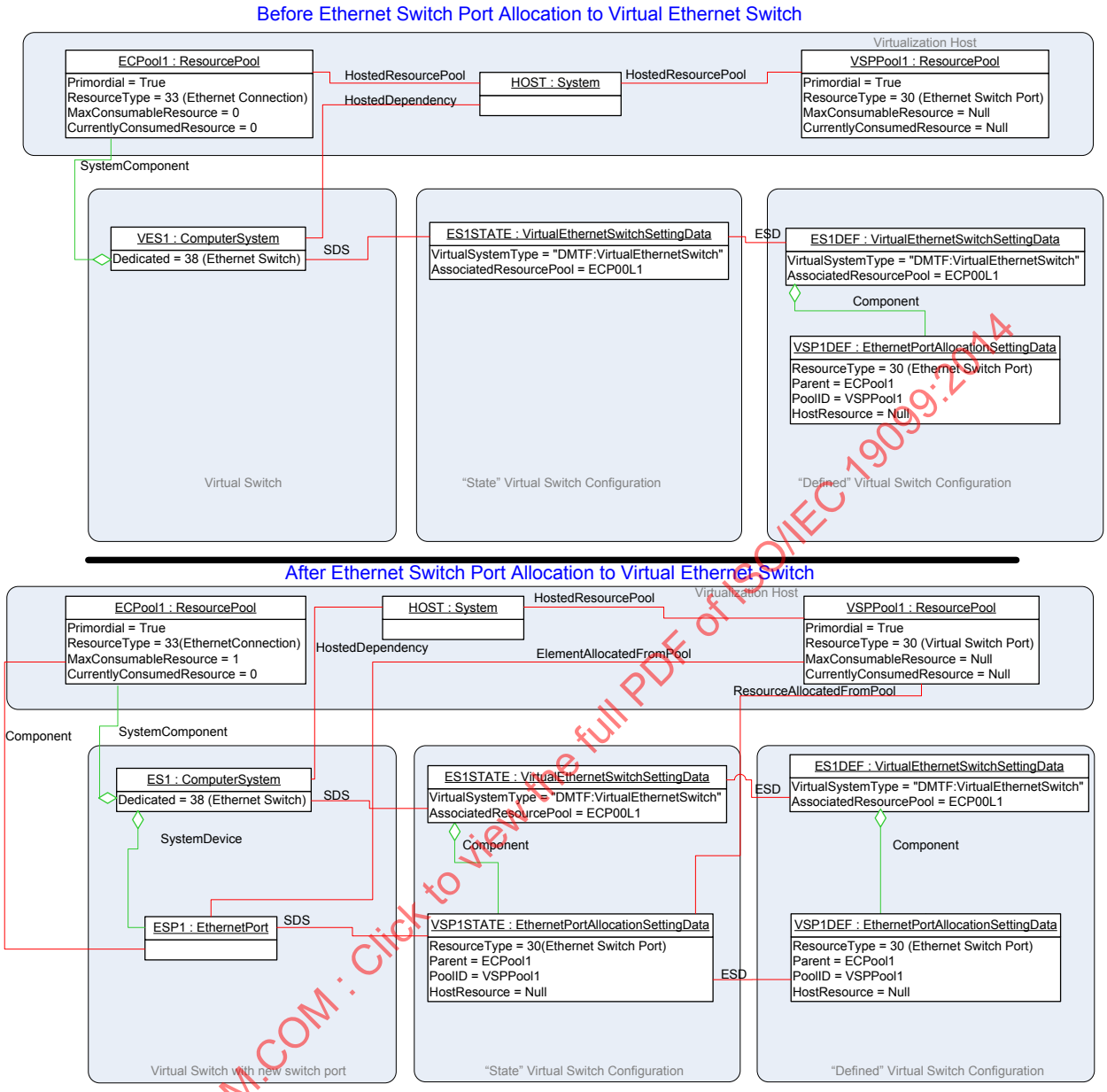

Figure 42 – Virtual ethernet switch port allocation

In the example shown in Figure 42, the virtual Ethernet switch is represented by the CIM_ComputerSystem instance VES1, as specified in the Virtual Ethernet Switch Profile described in clause 14. Once allocated, an Ethernet switch port is represented by a CIM_EthernetPort instance and a related CIM_LANEndpoint instance that is associated through an instance of the CIM_DeviceSAPImplementation association and represents the provided LAN endpoint.

In the example shown in Figure 42, the CIM_EthernetPortAllocationSettingData instance VSP1DEF represents an allocation request of an Ethernet switch port (resource type 30 [Ethernet Switch Port]) from the resource pool represented by VSPPOOL1. The value of the Parent property in VSP1DEF identifies the Ethernet connection resource pool represented by ECPOOL1 to provide the connection at allocation time.

The result of the allocation is shown in the lower half of Figure 42. An Ethernet switch port represented by the CIM_EthernetPort instance ESP1 has been allocated from the resource pool represented by VSPPOOL1, as shown through the instance of the CIM_ElementAllocatedFromPool association. ESP1 is associated with the CIM_ResourcePool instance ECPOOL1 through an instance of the CIM_ConcreteComponent association. This association represents the availability of the switch port for the allocation of Ethernet connections from the pool. Notice also that the addition of an Ethernet switch port is reflected by incrementing the value of the MaxConsumableResource property.

11.1.4.3 Ethernet resource allocation for virtual systems

Figure 43 shows an example of the allocation of Ethernet resources to a virtual system. The upper part of of Figure 43 shows allocation requests for an Ethernet adapter and a related static Ethernet connection for a virtual system. The lower part of Figure 43 shows the virtual system with the allocated Ethernet adapter and the allocated Ethernet connection.

NOTE This is a typical example; however, it is possible to request only an Ethernet Connection and receive an implicitly allocated default Ethernet adapter as part of the Ethernet connection allocation. (See the use case for the simple connection of a virtual machine described in 11.4.1.5 and Figure 50.)

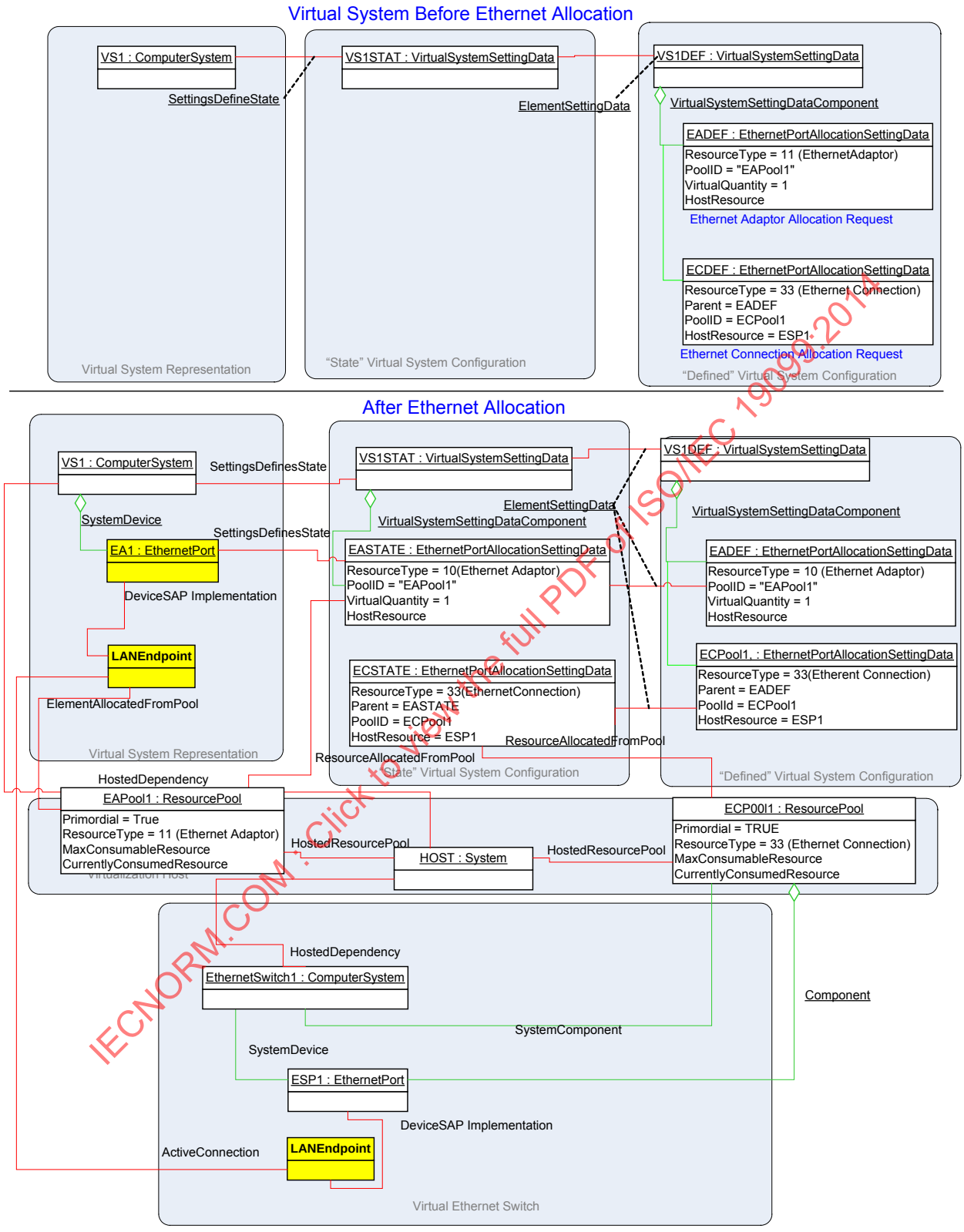

Figure 43 – Instance Diagram: Ethernet adapter and Ethernet connection resource allocations

11.1.4.4 Resource allocation request

The Ethernet connection and Ethernet adapter requirements of a virtual system are defined as part of its "defined" virtual system configuration; see the Virtual System Profile described in clause 12 for the specification of the "defined" virtual system configuration.

The "defined" virtual system configuration of a virtual system contains one or both of the following:

- Ethernet adapter resource allocation requests represented as EASD instances with the value of the ResourceType property set to 10 (Ethernet Adapter)
- Ethernet connection resource allocation requests represented as EASD instances with the value of the ResourceType property set to 33 (Ethernet Connection)

An example of the CIM representation of an Ethernet Adapter resource allocation request and a related Ethernet Connection resource allocation request is shown in the upper right part of Figure 43.

The Ethernet switch port requirements of a virtual system switch are defined as part of its "defined" virtual system configuration; see the Virtual Ethernet Switch Profile described in clause 14 for the specification of the "defined" virtual system configuration of virtual Ethernet switches.

The "defined" virtual system configuration of a virtual Ethernet switch contains Ethernet switch port resource allocation requests represented as EASD instances with the value of the ResourceType property set to 30 (Ethernet Switch Port).

An example of the CIM representation of an Ethernet switch port resource allocation request is shown in the upper right part of Figure 42.

11.1.4.5 Resource allocation

As a virtual system is activated (or instantiated), Ethernet adapters and Ethernet connections need to be allocated as requested by Ethernet adapter and Ethernet connection resource allocation requests in the virtual system definition. These resource allocations are represented as EASD instances in the "state" virtual system configuration; see the Virtual System Profile described in clause 12 for the specification of the "state" virtual system configuration.

An example of the CIM representation of an Ethernet Adapter and Ethernet Connection resource allocation is shown in the center part of Figure 43.

As a virtual Ethernet switch is activated (or instantiated), Ethernet switch ports need to be allocated as requested by Ethernet port resource allocation requests in the virtual system definition. These resource allocations are represented as EASD instances in the "state" virtual system configuration; see the Virtual Ethernet Switch Profile described in clause 14 for the specification of the "state" virtual system configuration of virtual Ethernet switches.

An example of the CIM representation of an Ethernet switch port resource allocation is shown in the center part of Figure 42.

11.1.4.6 Virtual Ethernet adapter

A virtual Ethernet adapter is either the instantiation of the resources allocated from an Ethernet adapter resource pool or instantiated as a side effect of an Ethernet connection allocation. The Ethernet adapter is represented with an instance of CIM_EthernetPort associated to the virtual system with CIM_SystemDevice.

In the example shown in Figure 43, the virtual Ethernet adapter was allocated from EA_Pool1 and is represented by the CIM_EthernetPort instance EA1 as part of the virtual system (VS1) representation.

11.1.4.7 Ethernet connection

A virtual Ethernet connection is the instantiation of resources allocated from an Ethernet connection resource pool. The allocation represents an allocation to connect an Ethernet adapter to an Ethernet switch port. A virtual Ethernet connection is not exposed to a virtual system through a logical device; however, a virtual Ethernet connection is represented by an instance of the CIM_ActiveConnection association between the CIM_LANEndpoint instance implemented by an Ethernet adapter and the CIM_LANEndpoint instance implemented by an Ethernet switch port. An Ethernet connection allocation can represent the connection between specific Ethernet adapter and Ethernet switch port instances, or the allocation could include the instantiation of an Ethernet adapter and/or an instantiation of an Ethernet switch port as part of the Ethernet connection allocation.

An example of the CIM representation of an Ethernet connection allocation is shown by the CIM_ActiveConnection association between the two CIM_LANEndpoint instances in Figure 43.

11.1.4.8 Virtual Ethernet switch port

A virtual Ethernet switch port is the instantiation of resources allocated from an Ethernet switch port resource pool or instantiated as part of an Ethernet connection allocation. The Ethernet switch port is represented with an instance of CIM_EthernetPort and associated to the CIM_ComputerSystem instance representing the virtual Ethernet switch with CIM_SystemDevice.

In the example shown in Figure 42, an allocated Ethernet switch port is represented by the CIM_EthernetPort instance ESP1 as part of the virtual Ethernet switch representation.

11.2 Implementation

This subclause provides normative requirements related to the arrangement of instances and properties of instances for implementations of the profile described in this clause.

11.2.1 Common requirements

The CIM Schema descriptions for any referenced element and its sub-elements apply.

In references to properties of CIM classes that enumerate values, the numeric value is normative and the descriptive text following it in parentheses is informative. For example, in the statement "The value of the ConsumerVisibility property shall be 3 (Virtualized)", the value "3" is normative text and "(Virtualized)" is informative text.

11.2.2 Resource types

This subclause specifies the resource types that are addressed by the profile described in this clause.

The profile described in this clause may be implemented for the allocation of two principal resource types: *Ethernet ports* and *Ethernet connections*. An Ethernet port is an Ethernet connection endpoint. Ethernet ports are further distinguished as *Ethernet adapters* and *Ethernet switch ports*. Ethernet adapters are Ethernet ports within virtual systems, and Ethernet switch ports are Ethernet ports within virtual switches.

11.2.3 Host resources

This subclause specifies requirements for the representation of host resources.

11.2.3.1 Host Ethernet adapters

The implementation of the representation of host Ethernet adapters is optional.

If implemented, the provisions in this subclause apply.

Each host Ethernet adapter shall be represented by exactly one CIM_EthernetPort instance. The CIM_EthernetPort instance shall be associated with the CIM_System instance that represents the host system through an instance of the CIM_SystemDevice association.

11.2.3.2 Host Ethernet switch ports

The implementation of the representation of host Ethernet switch ports is optional.

If implemented, the provisions in this subclause apply.

Each host Ethernet switch port shall be represented by exactly one CIM_EthernetPort instance. The CIM_EthernetPort shall be associated with either the CIM_System instance that represents the host system or the CIM_ComputerSystem instance that represents a virtual Ethernet switch hosted by the host system through an instance of the CIM_SystemDevice association.

11.2.4 Resource pool management feature

The implementation of the resource pool management feature is optional.

If implemented, the specifications of the Resource Allocation Profile described in clause 5 apply; the profile described in this clause does not specify specializations or extensions of resource pool management beyond those defined by the Resource Allocation Profile described (clause 5).

11.2.5 Resource pools

This subclause adapts the CIM_ResourcePool class for the representation of Ethernet adapter resource pools, Ethernet switch port resource pools, and Ethernet connection resource pools.

11.2.5.1 ResourceType property

The value of the ResourceType property shall denote the type of resources that are provided by the resource pool, as follows:

- For resource pools supporting only the allocation of Ethernet adapters, the value of the ResourceType property shall be 10 (Ethernet Adapter).
- For resource pools supporting only the allocation of Ethernet switch ports, the value of the ResourceType property shall be 30 (Ethernet Switch Port).
- For resource pools supporting only the allocation of Ethernet connections, the value of the ResourceType property shall be 33 (Ethernet Connection).

11.2.5.2 ResourceSubtype property

The implementation of the ResourceSubtype property is optional.

If the ResourceSubtype property is implemented, the provisions in this subclause apply.

The value of the ResourceSubtype property shall designate a resource subtype. The format of the value shall be as follows: "<org-id>:<org-specific>". The <org-id> part shall identify the organization that defined the resource subtype value; the <org-specific> part shall uniquely identify a resource subtype within the set of subtypes defined by the respective organization.

11.2.5.3 AllocationUnits property

If the allocation of Ethernet ports or Ethernet connections is based on bandwidth, the value of the AllocationUnits property shall be set to “bits per second” or a multiple thereof. The AllocationUnits property is a programmatic unit as specified in Annex C of DSP0004.

If the allocation of Ethernet ports is implemented based on the number of passed-through Ethernet ports, the value of the AllocationUnits property shall be set to "count" (the count of passed-through host Ethernet ports).

If the allocation of Ethernet connections is implemented based on the number of Ethernet connections, the value of the AllocationUnits property shall be set to "count" (the count of Ethernet connections).

11.2.5.4 Reserved property

The implementation of the Reserved property is optional.

If the Reserved property is implemented, the following provisions apply:

- If the value of the AllocationUnits property is (a multiple of) “bits per second the value of the Reserved property shall reflect the amount of Ethernet bandwidth that is actually reserved from the resource pool.
- If the value of the AllocationUnits property is "count", the value of the Reserved property shall denote the number of host Ethernet ports or the number of Ethernet connections that are actually reserved from the resource pool.

11.2.5.5 Capacity property

The implementation of the Capacity property is conditional.

Condition: The aggregation of host Ethernet ports into Ethernet port resource pools is implemented; see 11.2.4.

If the Capacity property is implemented, the following provisions apply:

- If the value of the AllocationUnits property is (a multiple of) "bits per second" (see DSP0004), the value of the Capacity property shall reflect the maximum aggregate amount of Ethernet bandwidth represented by the resource pool. If the resource pool has unlimited capacity, the value of the Capacity property shall be set to the largest value supported by the uint64 datatype.
- If the value of the AllocationUnits property is "count", the value of the Capacity property shall reflect the maximum number of host Ethernet ports or the maximum number of Ethernet connections represented by the resource pool.

11.2.5.6 MaxConsumableResource property

The implementation of the MaxConsumableResource property is conditional.

Condition: The resource pool supports the direct or exclusive allocation of a finite number of host resources.

If implemented, the value of the MaxConsumableResource property shall reflect the total number of virtual Ethernet adapters, virtual Ethernet switch ports, or virtual Ethernet connections that can be allocated in total from a resource pool.

11.2.5.7 ConsumedResourceUnits property

The implementation of the ConsumedResourceUnits property is conditional.

Condition: The MaxConsumableResource property or the CurrentlyConsumedResource property is implemented.

If implemented, the value of the ConsumedResourceUnits property shall be set to "count".

11.2.5.8 CurrentlyConsumedResource property

The implementation of the CurrentlyConsumedResource property is conditional.

Condition: The MaxConsumableResource property is implemented.

If implemented, the value of the CurrentlyConsumableResource shall reflect the total number of virtual Ethernet adapters, virtual Ethernet switch ports or virtual Ethernet connections that are currently allocated from the resource pool.

11.2.5.9 Instance requirements

Each Ethernet port resource pool shall be represented by exactly one CIM_ResourcePool instance. The CIM_ResourcePool instance shall be associated with the CIM_System instance representing the system hosting the resource pool through an instance of the CIM_HostedPool association.

11.2.5.10 Resource aggregation feature

The implementation of the resource aggregation feature is conditional.

Condition: The resource pool management feature is implemented; see 8.2.7.

Granularity: If implemented, the resource aggregation feature may be separately supported for each resource pool.

The preferred feature discovery mechanism is to resolve the CIM_Component association from the CIM_ResourcePool instance to CIM_ManagedElement instances representing aggregated resources of the storage resource pool. If the resulting set of CIM_ManagedElement instances is not empty, the feature is supported.

NOTE If the result set is empty, the feature may still be supported, but no resources are aggregated at that point in time; however, if aggregated resources for a particular resource pool were ever exposed, then the feature is still supported even if at a later point in time no resources are aggregated.

11.2.6 Resource allocation

This subclause details requirements for the representation of resource allocation information.

11.2.6.1 General

NOTE The Resource Allocation Profile described in clause 5 specifies two alternatives for modeling resource allocation: *simple resource allocation* and *virtual resource allocation*.

Implementations of the profile described in this clause shall implement the *virtual resource allocation* pattern as defined in 5.2.2 for resource types 10 (Ethernet Adapter) and 30 (Ethernet Switch Port).

Implementations of the profile described in this clause shall implement the *simple resource allocation* pattern as defined in 5.2.3 for resource types 33 (Ethernet Connection).

11.2.6.2 Flavors of allocation settings data

Details about the various flavors of allocation settings data are provided as follows:

- Resource allocation requests are described in 11.1.4.4.
- Resource allocations are described in 11.1.4.5.
- Settings that define the capabilities or mutability of managed resources are described in the Allocation Capabilities Profile described in clause 7, which specifies a capabilities model that

conveys information about the capabilities and the mutability of managed resources in terms of RASD instances (or instances of subclasses of RASD such as EASD).

- Parameters in operations that define or modify any of the previous representations in this list are described in the System Virtualization Profile (see clause 6), which specifies methods for the definition and modification of virtual resources. These methods use RASD instances (or instances of subclasses of RASD, such as EASD) for the parameterization of resource-allocation-specific properties.

Table 170 lists acronyms that are used in subclauses of 11.2.6 in order to designate EASD instances that represent various flavors of allocation settings data.

Table 170 – Acronyms for EASD adapted for the representation of various flavors of allocation data

Acronym	Flavor
Q_EASD	EASD adapted for the representation of Ethernet adapter resource allocation requests, Ethernet switch port resource allocation requests, or Ethernet connection resource allocation requests
R_EASD	EASD adapted for the representation of Ethernet adapter resource allocations, Ethernet switch port resource allocations, or Ethernet connection resource allocations
C_EASD	• EASD adapted for the representation of settings that define capabilities of systems or resource pools for Ethernet adapter resources, or that define the mutability of Ethernet adapter resource allocations or Ethernet adapter resource allocation requests • EASD adapted for the representation of settings that define capabilities of systems or resource pools for Ethernet switch port resources, or that define the mutability of Ethernet switch port allocations or of Ethernet switch port allocation requests • EASD adapted for the representation of settings that define capabilities of systems or resource pools, or that define the mutability of Ethernet connection resource allocations or Ethernet connection resource allocation requests
D_EASD	EASD adapted for the representation of new Ethernet adapter resource allocation requests in method parameter values, new Ethernet switch port resource allocation requests in method parameter values, or new Ethernet connection resource allocation requests in method parameter values as defined in the System Virtualization Profile described in clause 6
M_EASD	EASD adapted for the representation of modified Ethernet adapter resource allocations or Ethernet adapter resource allocation requests, EASD adapted for the representation of modified Ethernet switch port resource allocations or Ethernet switch port resource allocation requests, or EASD adapted for the representation of modified Ethernet connection resource allocations or Ethernet connection resource allocation requests in method parameter values as defined in the System Virtualization Profile described in clause 6

Subclauses of 11.2.6 detail implementation requirements for property values in EASD instances. In some cases requirements apply to only a subset of the flavors listed in Table 170; this is marked in the text through the use of respective acronyms.

11.2.6.2.1 CIM_EthernetPortAllocationSettingData properties

This subclause defines rules for values of properties in instances of the CIM_EthernetPortAllocationSettingData (EASD) class representing Ethernet port and Ethernet conection allocation information.

11.2.6.2.1.1 ResourceType property

The value of the ResourceType property shall denote the type of resources that are provided by the resource pool, as follows:

- For resource pools supporting the allocation of Ethernet adapters, the value of the ResourceType property shall be 10 (Ethernet Adapter).
- For resource pools supporting the allocation of Ethernet switch ports, the value of the ResourceType property shall be 30 (Ethernet Switch Port).
- For resource pools supporting the allocation of Ethernet connections, the value of the ResourceType property shall be 33 (Ethernet Connection).

11.2.6.2.1.2 ResourceSubType property

The implementation of the ResourceSubType property is optional.

If the ResourceSubType property is implemented, the provisions in this subclause apply.

The value of the ResourceSubType property shall designate a resource subtype. The format of the value shall be as follows: "<org-id>:<org-specific>". The <org-id> part shall identify the organization that defined the resource subtype value; the <org-specific> part shall uniquely identify a resource subtype within the set of subtypes defined by the respective organization.

11.2.6.2.1.3 PoolID property

The value of the PoolID property shall identify the current or desired resource pool. The special value NULL shall indicate the use of the host system's default resource pool for the selected resource type.

11.2.6.2.1.4 ConsumerVisibility property

The value of the ConsumerVisibility property shall denote either if a host resource is directly passed through to the virtual system as a virtual resource, or if the resource is virtualized. Values shall be set as follows:

- A value of 2 (Passed-Through) shall denote that the host resource is passed-through.
- A value of 3 (Virtualized) shall denote that the virtual resource is virtualized.
- Only in instances of { Q_RASD | D_RASD | M_RASD }, the special value NULL shall be used if the represented resource allocation request does not predefine which kind of consumer visibility (passed-through or virtualized) is requested.

Other values shall not be used.

11.2.6.2.1.5 AllocationUnits property

The value of the AllocationUnits property shall be set according to the rules defined in 11.2.5.3.

NOTE The units defined by the value of the AllocationUnits property apply to the values of the Reserved and Limit properties; the AllocationUnits property does not apply to the value of the VirtualQuantity property.

11.2.6.2.1.6 HostResource[ ] array property

The implementation of the HostResource[ ] array property is conditional.

Condition: One of the following:

- The implementation of the ResourceType property supports the value 33 (Ethernet Connection).
- The implementation of the ResourceType property supports the values 10 (Ethernet Adapter) or 30 (Ethernet Switch Port), together with values 3 (Dedicated), 4 (Soft Affinity), or 5 (Hard Affinity) for the MappingBehavior property.

If the HostResource[ ] array property is implemented, the provisions in this subclause apply.

If the value of the ResourceType property is 33 (Ethernet Connection), the value of the HostResource[ ] array property shall refer to one of the following:

- Exactly one CIM_EthernetPort instance that represents a specific target Ethernet switch port
- Exactly one CIM_ComputerSystem instance that represents a specific target Ethernet switch

If the value of the ResourceType property is 10 (Ethernet Adapter) or 30 (Ethernet switch port), in the cases of Q_EASD, C_EASD or M_EASD the following provisions apply:

- If the value of the MappingBehavior property is 3 (Dedicated), the value of the HostResource[ ] array property shall refer to one or more CIM_EthernetPort instances that represent host Ethernet adapter(s) or Ethernet switch port(s) that are exclusively dedicated to the virtual system or the virtual switch, respectively.
- If the value of the MappingBehavior property is 4 (Soft Affinity), the value of the HostResource[ ] array property shall refer to one or more CIM_EthernetPort instances that represent Ethernet adapter(s) or Ethernet switch port(s) preferably to be used for the allocation of the virtual Ethernet adapter or virtual Ethernet switch port.
- If the value of the MappingBehavior property is 5 (Hard Affinity), the value of the HostResource[ ] array property shall refer to one or more CIM_EthernetPort instances that represent Ethernet adapter(s) or Ethernet switch port(s) exclusively to be used for the allocation of the virtual Ethernet adapter or virtual Ethernet switch port.

If the value of the ResourceType property is 10 (Ethernet Adapter) or 30 (Ethernet switch port), in the cases of R_EASD, C_EASD or M_EASD the following provisions apply:

- If the value of the MappingBehavior property is 3 (Dedicated), 4 (Soft Affinity), or 5 (Hard Affinity), the value of the HostResource[ ] array property shall refer to one or more CIM_EthernetPort instances that represent a host Ethernet adapter or a host Ethernet switch port that support the allocated virtual Ethernet adapter or virtual Ethernet switch port.

11.2.6.2.1.7 VirtualQuantity property

If the value of the ResourceProperty is 10 (Ethernet Adapter) or 30 (Ethernet Switch Port), then the value of the VirtualQauntity property shall be the "count" of virtual Ethernet adapters or virtual Ethernet switch ports that are requested (in the cases of Q_EASD, D_EASD or M_EASD), allocated (in the case of R_EASD), or allowed (in the case of C_EASD).

If the value of the ResourceProperty is 33 (Ethernet Connection), then the value of the VirtualQauntity property shall be the "count" of virtual Ethernet connections that are requested (in the cases of Q_EASD, D_EASD or M_EASD), allocated (in the case of R_EASD), or allowed (in the case of C_EASD).

11.2.6.2.1.8 VirtualQuantityUnits property

The VirtualQuantityUnits property shall be set to “count”.

11.2.6.2.1.9 Reservation property

The implementation of the Reservation property is optional.

If the Reservation property is implemented, the value of the Reservation property shall denote the reserved amount; a requested reserve or a supported reserve amount of Ethernet bandwidth; or the count of Ethernet switch ports, Ethernet adapters, or Ethernet connections requested or supported in units of AllocationUnits.

If the Reservation property is not supported, it shall have a value of NULL. This indicates that an amount of host Ethernet bandwidth reserved for the use of the virtual system is not defined.

11.2.6.2.1.10 Limit property

The implementation of the Limit property is optional.

If the Limit property is implemented, the value of the Limit property shall denote either the maximum amount of Ethernet bandwidth available or the count of Ethernet switch ports, Ethernet adapters, or Ethernet connections requested or supported with regard to a virtual system in units of AllocationUnits.

The special value NULL shall indicate that a limit is not imposed.

11.2.6.2.1.11 Weight property

The implementation of the Weight property is optional.

If the Weight property is implemented, its value shall denote the relative priority of a resource allocation in relation to other resource allocations from the same pool.

The special value NULL shall indicate that a relative priority does not apply.

11.2.6.2.1.12 Parent property

The implementation of the Parent property is optional.

If the Parent property is implemented, the provisions in this subclause apply.

If the value of the ResourceType property value is 10 (Ethernet Adapter), the value of the Parent property shall refer to the parent entity of the resource allocation, or shall be NULL. The special value NULL shall indicate that a parent entity of the resource allocation is not defined.

If the value of the ResourceType property is 30 (Ethernet Switch Port) the following provisions apply:

- The Parent property may reference the desired, requested, allocated or allowed Ethernet connection resource pool that the allocated Ethernet switch port should be associated to with the CIM_ConcreteComponent association. The non-Null value of the Parent property shall conform to the production WBEM_URI_UntypedInstancePath as defined in DSP0207.

If the ResourceType property is 33 (Ethernet Connection), the following rules apply:

- Q_EASD: If the Parent property is Null, on allocation the provider shall instantiate an instance of CIM_EthernetPort and any associated LAN and VLAN endpoints representing an Ethernet adapter to the associated virtual machine and an R_EASD instance with the ResourceType property value set 33 (Ethernet Connection). This R_EASD instance and the instantiated instance of CIM_EthernetPort shall be associated through an instance of CIM_ElementSettingData.

Q_EASD: If the Parent property is not set to Null, then it shall specify an existing instance of an Ethernet adapter Q_EASD. On allocation the provider shall instantiate an R_EASD instance with the ResourceType property set 33 (Ethernet Connection) with its Parent property denoting the corresponding allocated Ethernet Adapter R_EASD instance. Each non-Null value of the Parent property shall conform to the production WBEM_URI_UntypedInstancePath as defined in DSP0207.

- D_EASD: The parent property may contain a temporary ID string that is correlated to a temporary ID string in the InstanceID property of a separate instance of D_EASD, where the ResourceType property is 10 (EthernetAdapter), instantiated as embedded instances in the same ResourceSettings parameter of a CIM_VirtualizationManagementService AddResourceSettings or DefineSystem method call. In this case the provider, as a result of the successful execution of the described method call, shall set the Parent property of the resultant Ethernet connection Q_EASD instance Parent property to reference the resultant Ethernet

adapter Q_EASD instance. In this case the Parent property shall conform to the production WBEM_URI_UntypedInstancePath as defined in DSP0207.

- R_EASD: If the Parent property is not Null, then the value of the Parent property shall reference the R_EASD instance that represents the target virtual Ethernet Adapter. The non-Null value of the Parent property shall conform to the production WBEM_URI_UntypedInstancePath as defined in DSP0207.

11.2.6.2.1.13 Address property

The implementation of the Address property shall be mandatory for R_EASD adaptations of CIM_EthernetPortAllocationSettingData. In all other adaptations of CIM_EthernetPortAllocationSettingData the Address property is optional.

If the address property is implemented, the provision in this subclause applies. The value of the Address property shall expose an address of the allocated resource that can be seen by the software running in the virtual system (usually the guest operating system). That address shall be unique at least within each resource type of a virtual system. That address may change over the lifetime of the allocated resource. A non-null value in the address property shall represent an Ethernet port identifier, most often the MAC_Address of the port.

If the ResourceType property is 10 (Ethernet Adapter), then a non-null value of the Address property shall contain an Ethernet port identifier (usually the MAC_Address) for a requested, defined, or allocated Ethernet Adapter.

If the ResourceType property is 30 (Ethernet Switch Port), then a non-null value of the Address property shall contain an Ethernet port identifier (usually the MAC_Address) for a requested, defined, or allocated Ethernet switch port.

If the ResourceType property is 33 (Ethernet Connection), then a non-null value of the Address property shall contain a network port identifier (usually the MAC_Address) for the target switch port.

The following rules apply:

- Q_EASD: If the Address property is Null, on allocation the provider shall provide a unique port identifier in the Address property of the R_EASD instance that is instantiated as a result of the allocation. If the parent property is not null the provider shall use the value in the Address property to set the Address property in the R_EASD instance that is instantiated as a result of the allocation.
- R_EASD: The value of the Address property shall reference the network port identifier of the target EthernetPort representing a virtual Ethernet adapter or virtual Ethernet switch.
- D_EASD, M_EASD: A non-null value of the Address property shall contain a string that is the requested network port identifier for an Ethernet adapter, Ethernet switch port, or connection to an Ethernet switch port.

11.2.6.2.1.14 InstanceID property

If CIM_EthernetPortAllocationSettingData property matches 10 (Ethernet Adapter), the following rule applies:

D_EASD: The InstanceID property may contain a temporary ID string that is correlated to a temporary ID string in the Parent property of a separate instance of D_EASD where the ResourceType property is 33 (Ethernet Connection), instantiated as embedded instances in the same ResourceSettings parameter of a CIM_VirtualizationManagementService AddResourceSettings or DefineSystem method call.

NOTE The D_EASD only exists as an embedded instance in a CIM_VirtualizationManagementService AddResourceSettings or DefineSystem method call.

11.2.6.2.1.15 Connection [ ] array property

The implementation of the Connection[ ] array property is optional.

If the Connection[ ] array property is implemented and the ResourceType property is set to 30 (Ethernet Switch Port) or 33 (Ethernet Connection), its value shall identify one or mode VLANs through their VLANIDs. The Connection[ ] array property shall contain exactly one VLANID if the value of the DesiredVLANEndPointMode property is 2 (Access). The Connection[ ] array property shall contain zero or more VLANIDs if the value of the DesiredVLANEndPointMode property is 5 (Trunk).

11.2.6.2.1.16 MappingBehavior property

The implementation of the MappingBehavior property is optional.

If the MappingBehavior property is implemented, its value shall denote how host resources referenced by elements in the value of HostResource[ ] array property relate to the Ethernet port resource allocation. The following rules apply:

- R_EASD only:
  - A value of 3 (Dedicated) shall indicate that the represented resource allocation is provided by host resources, as referenced by the value of the HostResource[ ] array property, that are exclusively dedicated to the virtual system.
  - A value of 4 (Soft Affinity) or 5 (Hard Affinity) shall indicate that the represented resource allocation is provided using the host EthernetPort resource as referenced by the value of the HostResource[ ] array property.
  - Other values shall not be used.
- Q_EASD, D_EASD, M_EASD only:
  - A value of 0 (Unknown) shall indicate that the resource allocation request or modification does not require specific host resources.
  - A value of 3 (Dedicated) shall indicate that the resource allocation request or modification shall be provided by exclusively dedicated host resources as specified through the value of the HostResource[ ] array property.
  - A value of 4 (Soft Affinity) shall indicate that the resource allocation request or modification shall preferably be provided by host resources as specified through the value of the HostResource[ ] array property, but that other resources may be used if the requested resources are not available.
  - A value of 5 (Hard Affinity) shall indicate that the resource allocation request or modification shall preferably be provided by host resources as specified through the value of the HostResource[ ] array property and that other resources shall not be used if the requested resources are not available.
  - Other values shall not be used.

The special value NULL shall indicate that a further qualification of the value of the HostResource[ ] array property through the value of the MappingBehavior property is not defined.

11.2.6.2.2 Instance requirements

This subclause details resource allocation-related instance requirements.

11.2.6.2.2.1 Representation of resource allocation requests

Each Ethernet adapter resource allocation request shall be represented by a Q_EASD instance; the provisions of 11.5.9 apply.

Each Ethernet switch port resource allocation request shall be represented by a Q_EASD instance; the provisions of 11.5.19 apply.

Each Ethernet connection resource allocation request shall be represented by a Q_EASD instance; the provisions of 11.5.14 apply.

11.2.6.2.2.2 Representation of resource allocations

Each Ethernet adapter resource allocation shall be represented by an R_EASD instance; the provisions of 11.5.10 apply.

Each Ethernet switch port resource allocation shall be represented by an R_EASD instance; the provisions of 11.5.20 apply.

Each Ethernet connection resource allocation shall be represented by an R_EASD instance; the provisions of 11.5.15 apply.

The R_EASD instance shall be associated to the Q_EASD instance representing the corresponding resource allocation request (see 11.1.4.4) through an instance of the CIM_ElementSettingData association; the provisions of 11.5.5 apply.

The R_EASD instance shall be associated to the CIM_ResourcePool instance providing resources for the allocation (see 11.2.5) through an instance of the CIM_ResourceAllocationFromPool association; see the Resource Allocation Profile described in clause 5.

Implementations may represent a resource allocation request and the corresponding resource allocation by one EASD instance; in this case, the association requirements of this subclause apply correspondingly. Association instances that refer to the A_EASD instance are only existent while the resource is allocated.

11.2.6.2.2.3 Representation of resource allocation capabilities

The allocation capabilities of a system or a resource pool shall be represented by a CIM_AllocationCapabilities instance that is associated to the CIM_System instance representing the system or to the CIM_ResourcePool instance representing the resource pool through an instance of the CIM_ElementCapabilities association; see the Allocation Capabilities Profile described in clause 7.

The settings that define the Ethernet adapter allocation capabilities of an Ethernet adapter resource pool or of a system shall be represented by C_EASD instances; the provisions of 11.5.11 apply.

The settings that define the Ethernet switch port allocation capabilities of an Ethernet switch port resource pool or of a system shall be represented by C_EASD instances; the provisions of 11.5.21 apply.

The settings that define the Ethernet connection allocation capabilities of an Ethernet connection resource pool or of a system shall be represented by C_EASD instances; the provisions of 11.5.16 apply.

11.2.6.2.2.4 Representation of resource allocation mutability

The mutability of a resource allocation or resource allocation request shall be represented by a CIM_AllocationCapabilities instance that is associated to the EASD instance representing the resource allocation or resource allocation request through an instance of the CIM_ElementCapabilities association; see the Allocation Capabilities Profile described in clause 7.

The settings that define the allocation mutability of an Ethernet adapter resource allocation or an Ethernet adapter resource allocation request shall be represented by C_EASD instances; the provisions of 11.5.11 apply.

The settings that define the allocation mutability of an Ethernet switch port resource allocation or an Ethernet switch port resource allocation request shall be represented by C_EASD instances; the provisions of 11.5.21 apply.

The settings that define the allocation mutability of an Ethernet connection resource allocation or an Ethernet connection resource allocation request shall be represented by C_EASD instances; the provisions of 11.5.16 apply.

11.2.7 Virtual resources

11.2.7.1 Virtual Ethernet adapter

Each allocated virtual Ethernet adapter shall be represented by one CIM_EthernetPort instance that is associated with the CIM_ComputerSystem instance that represents the virtual system through an instance of the CIM_SystemDevice association; the provisions of 11.5.29 apply.

The CIM_EthernetPort instance shall be associated with the CIM_ResourcePool instance from which it was allocated through the CIM_ElementAllocatedFromPool association; the provisions of 11.5.3 apply.

Each connection endpoint implemented by the Ethernet adapter shall be represented by a CIM_LanEndpoint instance that is associated to the CIM_EthernetPort instance through an instance of the CIM_DeviceSAPImplementation association as specified in DSP1014.

NOTE The profile described in this clause does not attempt to specify the mapping of the characteristics or the implementation of the physical characteristics mandated by the dependency on DSP1014. For example, there are no physical characteristics or bandwidth requirements mandated by this specification to allow a provider to set the PortType property of CIM_EthernetPort to “1000BaseT”.

11.2.7.2 Virtual Ethernet switch port

Each allocated virtual Ethernet switch port shall be represented by one CIM_EthernetPort instance that is associated with the CIM_ComputerSystem instance that represents the virtual Ethernet switch through an instance of the CIM_SystemDevice association; the provisions of 11.5.29 apply.

The CIM_EthernetPort instance shall be associated with the CIM_ResourcePool instance from which it was allocated through the CIM_ElementAllocatedFromPool association; the provisions of 11.5.3 apply.

Each connection endpoint implemented by the Ethernet switch port shall be represented by a CIM_LANEndpoint instance that is associated to the CIM_EthernetPort instance through an instance of the CIM_DeviceSAPImplementation association as specified in DSP1014.

11.2.7.3 Virtual Ethernet connection

Each virtual Ethernet connection resource allocation shall be represented by one instance of the CIM_ActiveConnection association that associates the CIM_LANEndpoint instances representing the connection endpoints that are associated to the targeted virtual Ethernet adapter (see 11.2.7.1) and virtual Ethernet switch port (see 11.2.7.2). The provisions of 11.5.1 apply.

The CIM_LANEndpoint instance associated to the CIM_EthernetPort instance representing the Ethernet adapter shall be associated with CIM_ElementSettingData to the R_EASD instance representing the allocated connection resources. The provisions of 11.5.4 apply.

11.3 Methods

This subclause details the requirements for supporting operations and methods for the CIM elements defined by the profile described in this clause.

11.3.1 Profile conventions for operations

The implementation requirements on operations for each profile class (including associations) are specified in class-specific subclauses of this subclause.

The default list of operations for all classes is:

- GetInstance( )
- EnumerateInstances( )
- EnumerateInstanceNames( )

For classes that are referenced by an association, the default list also includes

- Associators( )
- AssociatorNames( )
- References( )
- ReferenceNames( )

Implementation requirements on operations defined in the default list are provided in the class-specific subclauses of this subclause.

The implementation requirements for methods of classes listed in 11.5, but not addressed by a separate subclause of this subclause, are specified by the "Methods" clauses of respective base profiles, namely the Resource Allocation Profile described in clause 5 and the Allocation Capabilities Profile described in clause 7. These profiles are specialized by the Memory Resource Virtualization Profile, and in these cases this profile does not add method specifications beyond those defined in its base profiles.

11.3.2 CIM_EthernetPort for host systems

All operations in the default list in 11.3.1 shall be implemented as specified by DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

11.3.3 CIM_EthernetPort for virtual systems

All operations in the default list in 11.3.1 shall be implemented as specified by DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

11.3.4 CIM_EthernetPortAllocationSettingData

All operations in the default list in 11.3.1 shall be implemented as specified by DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

11.3.5 CIM_ResourcePool

All operations in the default list in 11.3.1 shall be implemented as specified by DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

11.3.6 CIM_SystemDevice for host resources

All operations in the default list in 11.3.1 shall be implemented as specified by DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

11.3.7 CIM_SystemDevice for virtual resources

All operations in the default list in 11.3.1 shall be implemented as specified by DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

11.3.8 CIM_VLANEndpointSettingData

All operations in the default list in 11.3.1 shall be implemented as specified by DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

11.4 Use cases

The use cases and object diagrams in this subclause illustrate use of the profile described in this clause. They are for informative purposes only and do not introduce behavioral requirements for implementations of the profile.

11.4.1 Instance diagrams

The following use cases represent three separate example implementation options of varying complexity:

- Static represents the fully featured allocation model. It defines Ethernet connection allocations to existing Ethernet switch port instances that are aggregated host resources into an Ethernet connection resource pool. This implementation option allows for the separate management of the Ethernet switch ports as part of the virtual Ethernet switch. In this option, there are resource pools for all three EthernetPortAllocationSettingData resource types: Ethernet Connection, Ethernet Adapter and Ethernet Switch Port. Ethernet connection allocations are used to connect to an existing Ethernet switch port and a defined Ethernet adapter. If allowed by the implementation, the relevant properties in the Ethernet Connection request are used to override the values set in the Ethernet switch port allocation.
- Dynamic simplifies the model by dynamically generating an Ethernet switch port instance on a virtual Ethernet switch at the time that the Ethernet connection allocation targeting a switch is made. Ethernet connection allocations are used to connect a defined Ethernet adapter to a dynamically allocated Ethernet switch port. If allowed by the implementation, the relevant properties in the Ethernet Connection request are used to override the default values for the corresponding settings in the Ethernet switch port.
- Simple further simplifies the model using only an Ethernet connection allocation to create a complete network connection. On the allocation of an Ethernet Connection to a virtual machine targeting a virtual Ethernet switch, both an Ethernet adapter and an Ethernet switch port are dynamically allocated. If allowed by the implementation, the relevant properties in the Ethernet Connection request are used to override the default values for the corresponding settings in the Ethernet switch port.

The preceding three example implementations are not presented as any limitation of possible implementations; rather they are illustrative of the target models that lead the development of the profile described in this clause.

11.4.1.1 Static Ethernet switch port and Ethernet connection resource pools with capabilities

Figure 44 is a CIM representation of a virtualization system (HOST) with a hosted virtual Ethernet switch (VSWITCH0) and resource pools for Ethernet switch ports (SP_POOL) and Ethernet connections

(EC_POOL). Figure 44 also has a set of capabilities for the two resource pools. The system as represented supports static switch port allocations to an Ethernet switch.

SP_POOL represents a resource pool of unlimited capabilities of allocating virtualized Ethernet switch ports with a desired mode of either Trunk or Access. These capabilities are shown through the CIM_AllocationCapabilities instance (CAP_ESP) and two instances of the CIM_EthernetPortAllocationSettingData class (CAP_POINT0 and CAP_POINT1), associated through two instances of the CIM_SettingDefinesCapabilities association class.

CAP_POINT1 is a default capabilities instance. The value of the CAP_POINT1 DesiredVLANEndpointMode property is set to 2 (Access). Only virtual instances of Ethernet switch ports are supported from this pool as represented by the value 2 (Virtualized) of the ConsumerVisibility property.

The value of the CAP_POINT0 DesiredVLANEndpointMode property is set to 5 (Trunk), indicating that a Trunking Ethernet switch port can also be allocated from the resource pool SP_POOL. Again, only virtual instances of Ethernet switch port allocations are supported from this pool, as represented by the value 2 (Virtualized) for the ConsumerVisibility property.

The virtual Ethernet switch represented by an instance, VSWITCH0, of the CIM_ComputerSystem class as shown in Figure 44 has one associated Ethernet connection resource pool represented by the EC_POOL instance of the CIM_ResourcePool class. EC_POOL represents a pool with 10 gigabits of bandwidth as shown by the value of the Capacity property (equal to 10,000 combined with the AllocationUnits property of “bits per second*2^20”). EC_POOL currently has no assigned Ethernet switch ports that are available for connection because the value of the MaxConsumableResource property is 0.

EC_POOL has an associated instance CAP_EC of the CIM_AllocationCapabilities class with a set of CIM_EthernetPortAllocationSettingData instances to describe the supported allocations from the pool when there are Ethernet switch ports available for connection. An examination of these instances of the CIM_EthernetPortAllocationSettingData class (CAP_EC_MIN, CAP_EC_MAX, CAP_EC_INC, CAP_EC_POINT0, and EC_POINT1) describe the capabilities of the EC_POOL resource pool:

- Only Dedicated allocations are allowed (MappingBehavior = 3 [Dedicated]) in all instances.
- The default allocation request is 1,000 megabytes of reserved bandwidth (Reserved = 1000) with 10,000 megabyte top limit of allowable bandwidth (Limit = 10000). The default allocation has VLAN support with the value of the DesiredVLANEndpointMode property set to “Access”. These values are shown in the CAP_EC_DEF instance of the CIM_EthernetPortAllocationSettingData class.
- The empty string value in the Parent property shows that the system supports the setting of the value of the Parent property, which is limited by the profile described in this clause to be a reference URI to the Ethernet adapter request instance of the CIM_EthernetPortAllocationSettingData class.
- Allocation request reservation and limit values can be made in the range of 100 to 10,000 megabits per second of bandwidth, with an increment of 100 megabits per second. This range is shown in the CAP_EC_MAX, CAP_EC_MIN, and CAP_EC_INC instances of the CIM_EthernetPortAllocationSettingData class Reservation and Limit property values.
- VLAN is supported, and either Access or Trunk mode is supported. (See the DesiredVLANEndPointMode property values for the CAP_EC_POINT0 and CAP_EC_POINT1 instances.)
- The array of supported VLANID is represented in the value of the Connection array properties in the CAP_EC_POINT0 and CAP_EC_POINT1 instances.

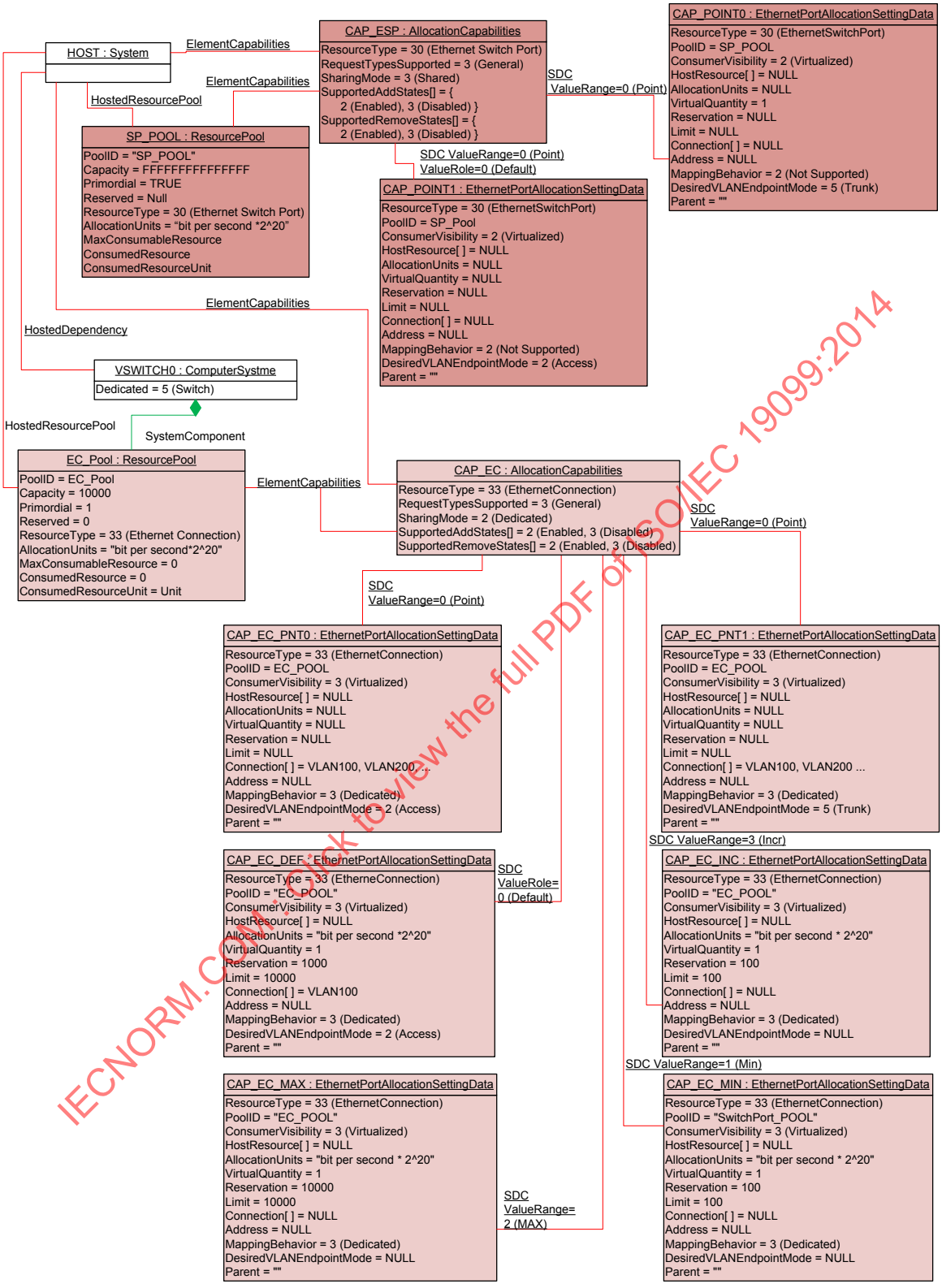

Figure 44 – Ethernet switch port and Ethernet connection resource pools

11.4.1.2 Static Ethernet switch port allocation to a virtual Ethernet switch

Figure 45 shows the same host system (HOST) and virtual Ethernet switch (VSWITCH0) as shown in Figure 44 with the resource pool allocation capabilities removed to simplify the drawing. Figure 45 is the CIM representation of the system after a static Ethernet switch port, represented as the instance ESP0 of the CIM_EthernetPort class, has been allocated to the virtual Ethernet switch VSWITCH0 from the instance of the host resource pool SP_POOL representing the CIM_ResourcePool class.

The allocation of ESP0 is a virtual resource allocation as described in the Resource Allocation Profile described in clause 5. Thus, it has an associated state instance of the CIM_EthernetPortAllocationSettingData class (ESAD_ESP0). In this use case this same instance is also used as the request instance, as shown with the self-reference of the CIM_ElementSettingData association to EASD_ESP0.

An examination of values in the properties of EASD_ESP0 shows that a default allocation was used in the allocation request because the DesiredVLANEndpointMode is set to Access. The provider in this use case provided a MAC address (MAC_ADDRESS) and inserted the default VLANID for the associated virtual Ethernet switch port into the Connection property.

Associated to EASD_ESP0 is a CIM_AllocationCapabilities instance (MUT_ESP). Associated to MUT_ESP are two mutability instances of CIM_EthernetPortAllocationSettingData (MUT_POINT0 and MUT_POINT1), which shows that the DesiredVLANEndPointMode and Connection properties are mutable. The DesiredVLANENDPointMode property value can be changed to either 2 (Access) or 5 (Trunk). The VLANID Access property can be set to any of the values listed in the Connection property of instance MUT_POINT1.

Because the Parent property value of instance EASD_ESP0 was set to reference the resource pool instance EC_POOL, the allocated CIM_EthernetPort instance ESP0 is included in the CIM_Component aggregation to the EC_POOL resource pool. Also note that the MaxConsumableResource property value has been incremented to 1 from the value shown in Figure 44 to show that a switch port is available for connection.

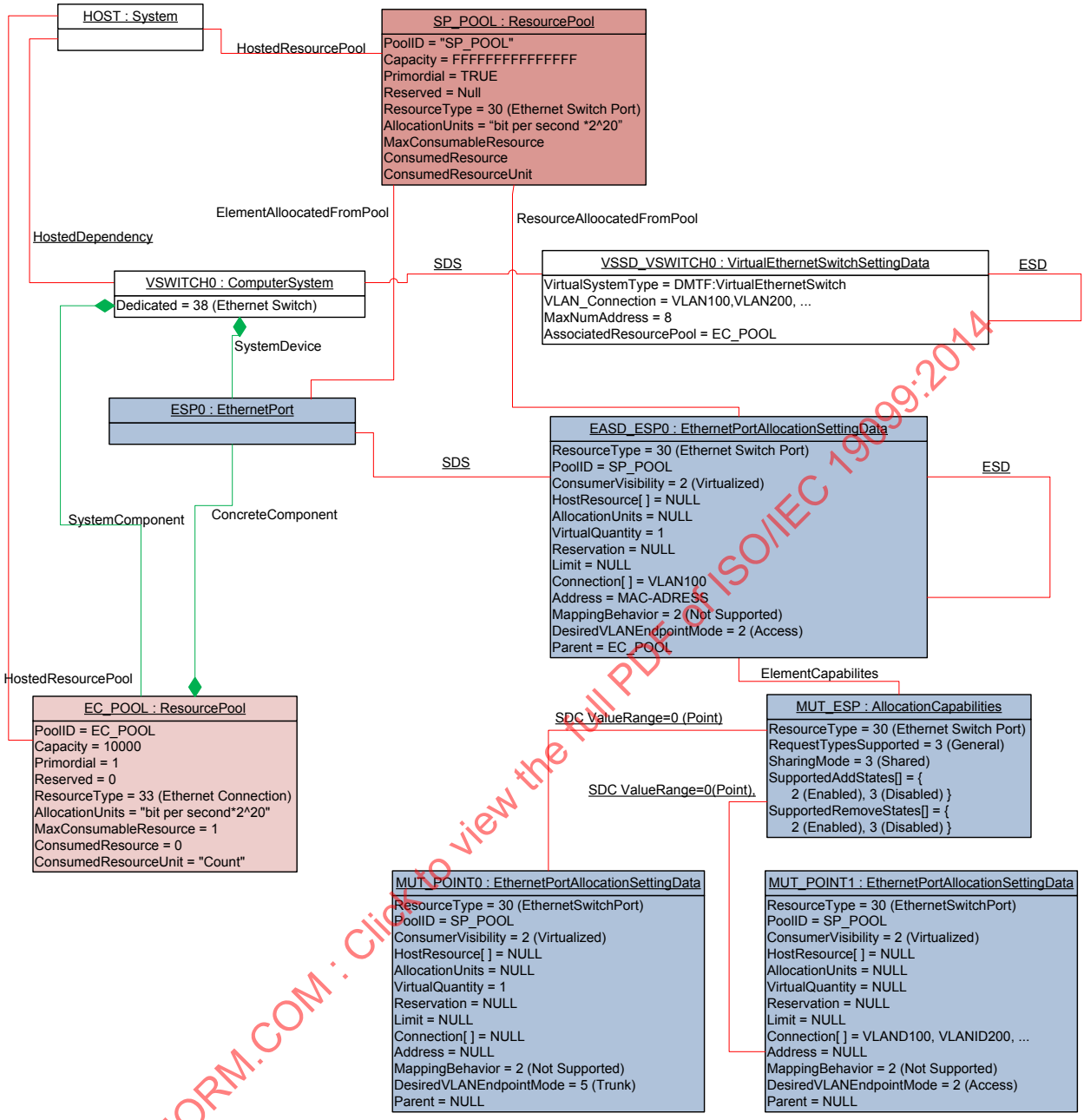

Figure 45 – Static Ethernet switch port allocation to a virtual Ethernet switch

11.4.1.3 Allocation and connection of an Ethernet adapter to a static switch port

Figure 46 shows the same virtualization system and virtual Ethernet switch shown in Figure 45 and Figure 44. This figure includes an instance of a virtual system (VM1) represented with the CIM_ComputerSystem class with allocation requests and a current device allocation for an Ethernet Adapter instance (EA) represented by the CIM_EthernetPort class and a simple allocation of an Ethernet connection to the Ethernet switch port ESP0. No allocation capabilities are shown in this figure, but the Allocation Capabilities for the Ethernet connection resource pool EC_POOL are as shown in Figure 44.

The Ethernet adapter request for VM1, the EA_REQ instance of the CIM_EthernetPortAllocationSettingData class, shows that this provider allows the allocation of synthetic Ethernet adapters with no host resource allocation. This is shown with the unlimited capacity of EA_POOL and the NULL values in the EA_REQ instance for the Reserve and Limit properties. This allocation is a basic virtual resource allocation with the purpose of allocating a logical device instance of the CIM_EthernetPort class. The provider populated the value in the Address property in the state instance (EA_STATE) of the CIM_EthernetPortAllocationSettingData class with a MAC address represented in Figure 46 as EA_MAC. The allocation is a virtual resource allocation as shown by the CIM_ElementAllocatedFromPool association between the resource pool EA_POOL and the EA instance of CIM_EthernetPort as well as the CIM_ResourceAllocatedFromPool association instance between EA_POOL and EA_State.

The Ethernet connection request for VM1, the EC_REQ instance of the CIM_EthernetPortAllocationSettingData class, specifies a request for a specific Ethernet switch port (ESP0), a reservation and limit of Bandwidth through the switch (VSWITCH0), and a set of VLAN property overrides of the default properties of the requested Ethernet switch port. The property values of EC_REQ define the request EASD as follows:

- PoolID=EC_POOL: This selects the resources pool EC_POOL.
- Parent=EA_REQ: This associates this Ethernet connection request with the Ethernet adapter request EA_REQ.
- HostResource[] = ESP0: This requests that specific Ethernet switch port.
- MappingBehavior = 3 (Dedicated): This property identifies that this is an exclusive request for this resource.
- AllocationUnits=bits per second*2^20: This specifies a bandwidth unit of 1 megabyte per second.
- Reservation=1000: This requests to reserve 1 gigabit per second of Ethernet bandwidth.
- Limit=10000: This sets a limit of 10 gigabits per second. In effect, there is no limit to the VM's use of available bandwidth because this value matches the maximum capacity of the request resource pool.
- Address=NULL: There is no request to override the MAC address of the switch port.
- DesiredVLANEndpointMode=Access: The request sets and maintains the desired VLANEndpointMode of the requested Ethernet switch port.
- Connection=VLAN200: This is an override of the access VLANID for the switch port.
- VirtualQuantity=1: This is a request for one connection.

The Ethernet connection state instance EC_STATE represents the current allocation of the Ethernet connection described above. The only property value difference between the EC_STATE and EC_REQ is the value of the Parent property. The value of the Parent property is a reference to the Ethernet adapter's allocation instance EA_STATE represented with the CIM_EthernetPortAllocationSettingData class.

When VM1 was turned on the Ethernet adapter (EA) and its associated CIM_LANEndpoint instance (EA_LEP) were instantiated based on the value of the request instance EA_REQ. Based on the Ethernet connection request instance (EC_REQ), the provider instantiated the Ethernet switch port's associated instance of CIM_LANEndpoint (ESP_LEP), the instance of CIM_VLANEndpoint(VLEP), and the instance of VLANEndpointSettingData(VEPSD). The property values shown in these instances are the corresponding properties described in the above description of EC_REQ.

The connection between the two CIM_LANEndpoint instances, EA_LEP and ESP_LEP, is shown with the association CIM_ActiveConnection.

The connection to the Ethernet switch port, ESP0, is noted with the incremented value of the EC_POOL ConsumedResource property from the value shown in Figure 45.

Lastly, the instantiated CIM_VLANEndpoint is associated to the corresponding VLAN200 instance of the CIM_NetworkVLAN class through a CIM_MemberOfCollection association.

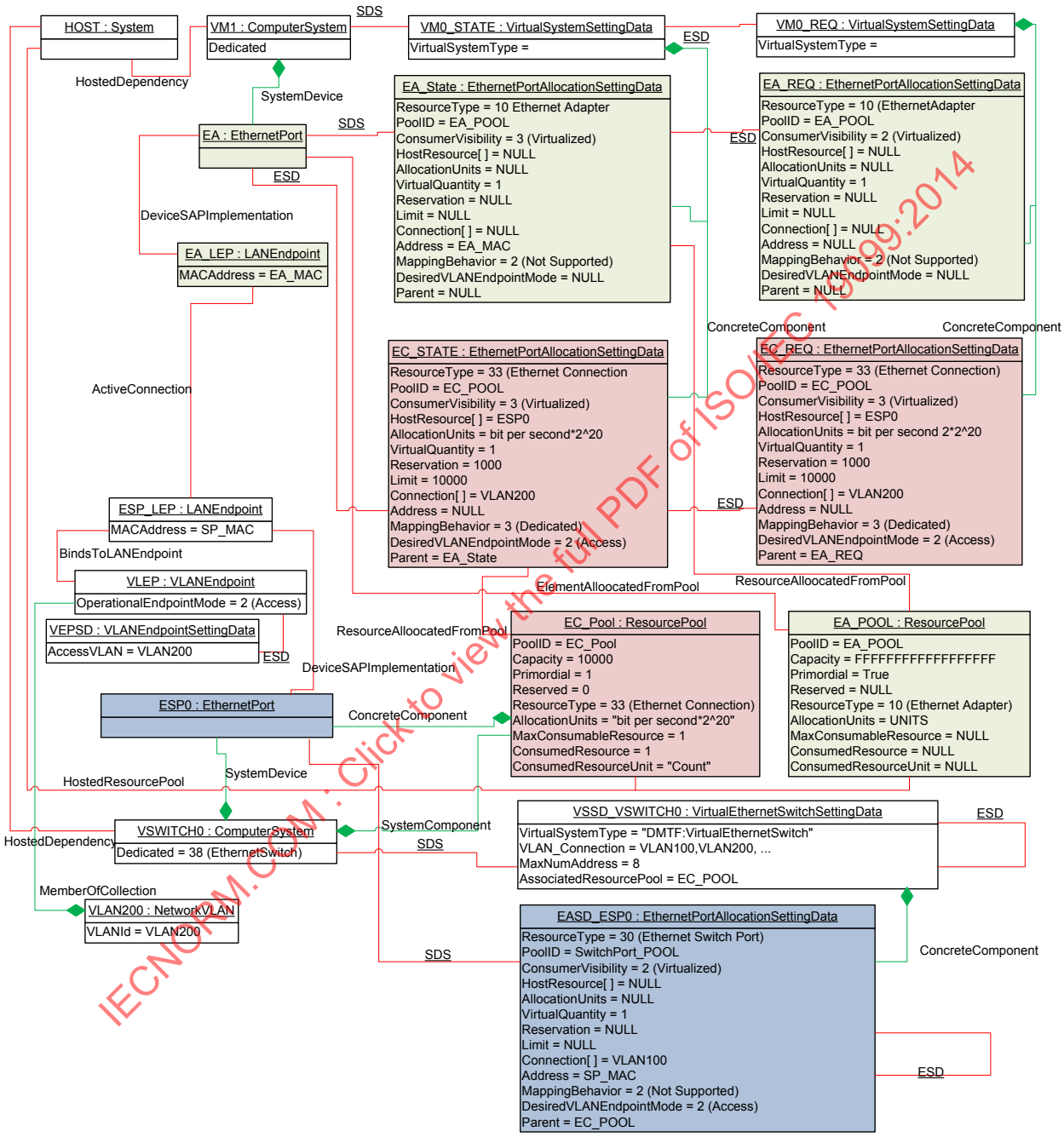

Figure 46 – Ethernet adapter connection to static switch port

11.4.1.4 Connection of an Ethernet adapter to an Ethernet switch (dynamic switch port allocation)

Figure 47 and Figure 48 are CIM instance diagrams that represent a virtualization system that supports dynamic or implied switch port allocation during the connection of an Ethernet adapter to a virtual Ethernet switch.

Figure 47 is a CIM representation of the allocation capabilities (CAP_EC) of an Ethernet connection resource pool (EC_POOL) associated with a virtual Ethernet switch (VSWITCH1).

The resource pool EC_POOL has a resource type of 33 (Ethernet Connection). The pool has a capacity of 10 gigabits of Ethernet bandwidth. This pool has no defined limits on the number of connections that can be made, as shown with NULL values for the MaxConsumableResource and ConsumedResource properties in EC_POOL.

The CIM_AllocationCapabilities instance CAP_EC has six associated instances of CIM_EthernetPortAllocationSettingData that are associated through the CIM_SettingDefinesCapabilities association:

- Instance CAP_EC_DEF shows that a connection to VSWITCH1 is made by requesting VSWITCH1 as a reference value in the HostResource property and EC_POOL in the PoolID property. This default request is a request for 1 gigabit of bandwidth as shown with a reserved property value of 1000 and the AllocationUnit property value of bits per second * 2^20. The default value for the DesiredVLANEndpointMode is Access with a VLANID of VLAN100. The empty string value in the Parent property shows that the system supports the modification ofthe Parent property. The use of the Parent property in this use is limited by the profile described in this clause to be a reference to the Ethernet adapter request instance of the CIM_EthernetPortAllocationSettingData class.
- Instances CAP_EC_INC, CAP_EC_MAX, and CAP_EC_MIN define the valid range of values for the Reserve and Limit properties and the Increment value for those properties.
- The values in the DesiredVLANEndpointMode property of the CAP_EC_PNT1 and CAP_EC_PNT0 capabilities instances show that either 2 (Access) or 5 (Trunk) can be requested. The values listed in the Connection property for both instances list the valid VLANIDs that can be requested in an allocation request.

Figure 48 shows the same virtualization system with a dynamic Ethernet connection allocation and an active Ethernet adapter allocation to VM1. The Ethernet adapter allocation is identical to the allocation shown in Figure 46 and described in 11.4.1.3.

The Ethernet connection request and allocation instances of CIM_EthernetPortAllocationSettingData (EC_REQ and EC_STATE) are for a dynamic Ethernet port allocation. As a side effect of the Ethernet connection allocation, an Ethernet switch port instance (ESP0), its associated LAN and VLAN endpoints (ESP_LEP and VLEP), and an instance of CIM_VLANEndpointSettingData (VEPSD) are instantiated.

The Ethernet connection request for VM1, the EC_REQ instance of the CIM_EthernetPortAllocationSettingData class, specifies a default Ethernet switch port from the virtual Ethernet switch VSWITCH0, a reservation and limit of bandwidth through the switch VSWITCH0, and a set of VLAN property values for the Ethernet switch port. The property values of EC_REQ define the request instance of EASD as follows:

- PoolID=EC_POOL: This selects the resource pool EC_POOL.
- Parent=EA_REQ: This associates this Ethernet connection request with the Ethernet adapter request EA_REQ.
- HostResource[] = VSWITCH1: This requests that an Ethernet switch port as defined by the allocation capabilities associated with the Ethernet connection resource pool EC_POOL be instantiated.

- MappingBehavior = 2 (Not Supported)
- AllocationUnits = bit per second*2^20: This specifies a bandwidth unit of 1 megabyte per second.
- Reservation = 1000: This is a request to reserve 1 gigabit per second of Ethernet bandwidth.
- Limit = 10000: This sets a limit of 10 gigabits per second; in effect, there is no limit to the VM's use of available bandwidth because this value matches the maximum capacity of the request resource pool.
- Address = NULL: There is no request to override the provider-generated MAC address of the switch port.
- DesiredVLANEndpointMode = Access: This requests the desired VLANEndpointMode of the requested Ethernet switch port.
- Connection = VLAN200: This requests the access VLANID for the switch port.
- VirtualQuantity = 1: This is a request for one connection.

The Ethernet connection state EASD (EC_STATE) represents the current allocation of the Ethernet connection described above. The only different property value from the instance EC_REQ in this use case is for the Parent property, which reflects the Ethernet adapter allocation EA_STATE instead of EA_REQ.

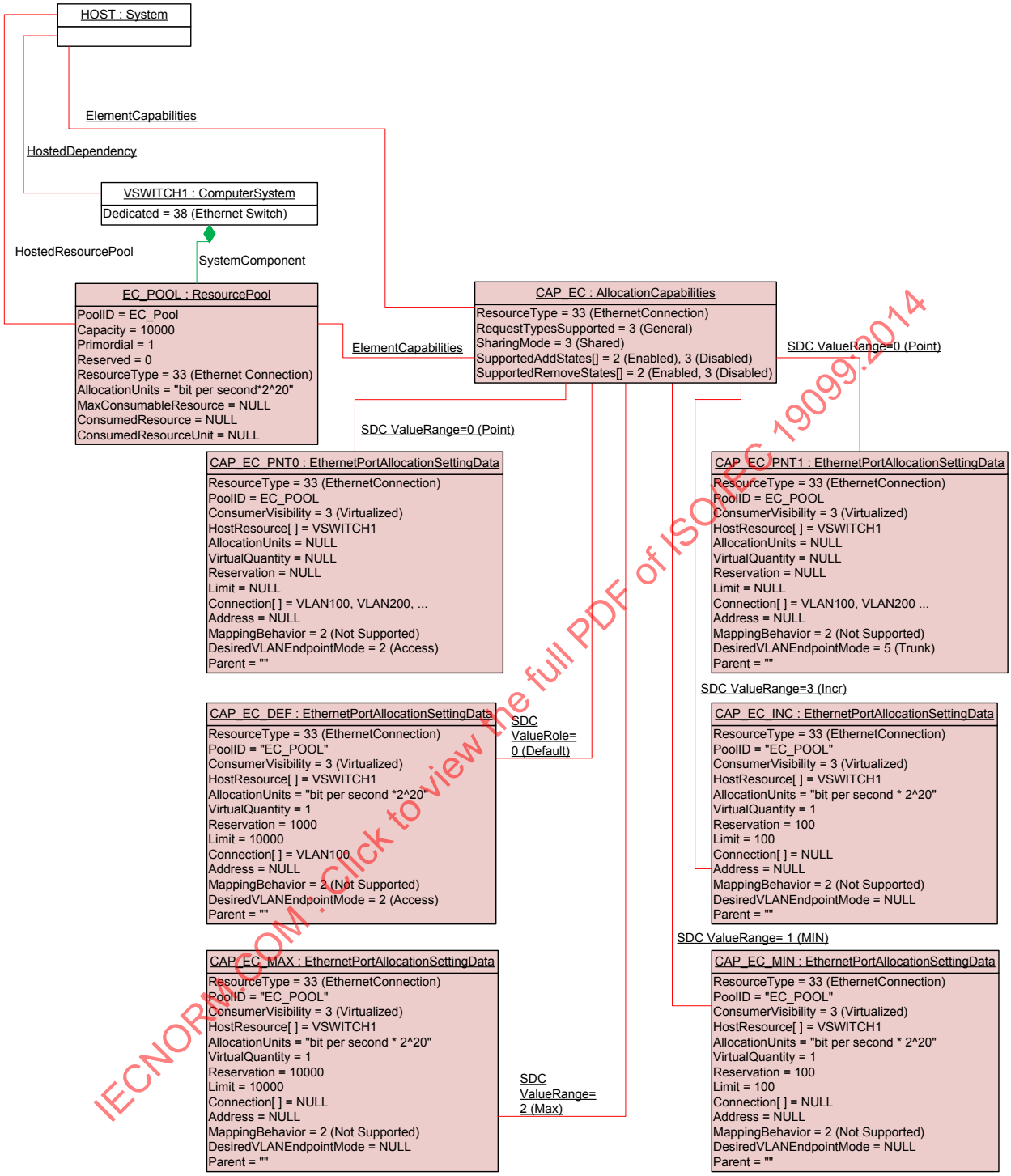

Figure 47 – Dynamic Ethernet switch port connection capabilities

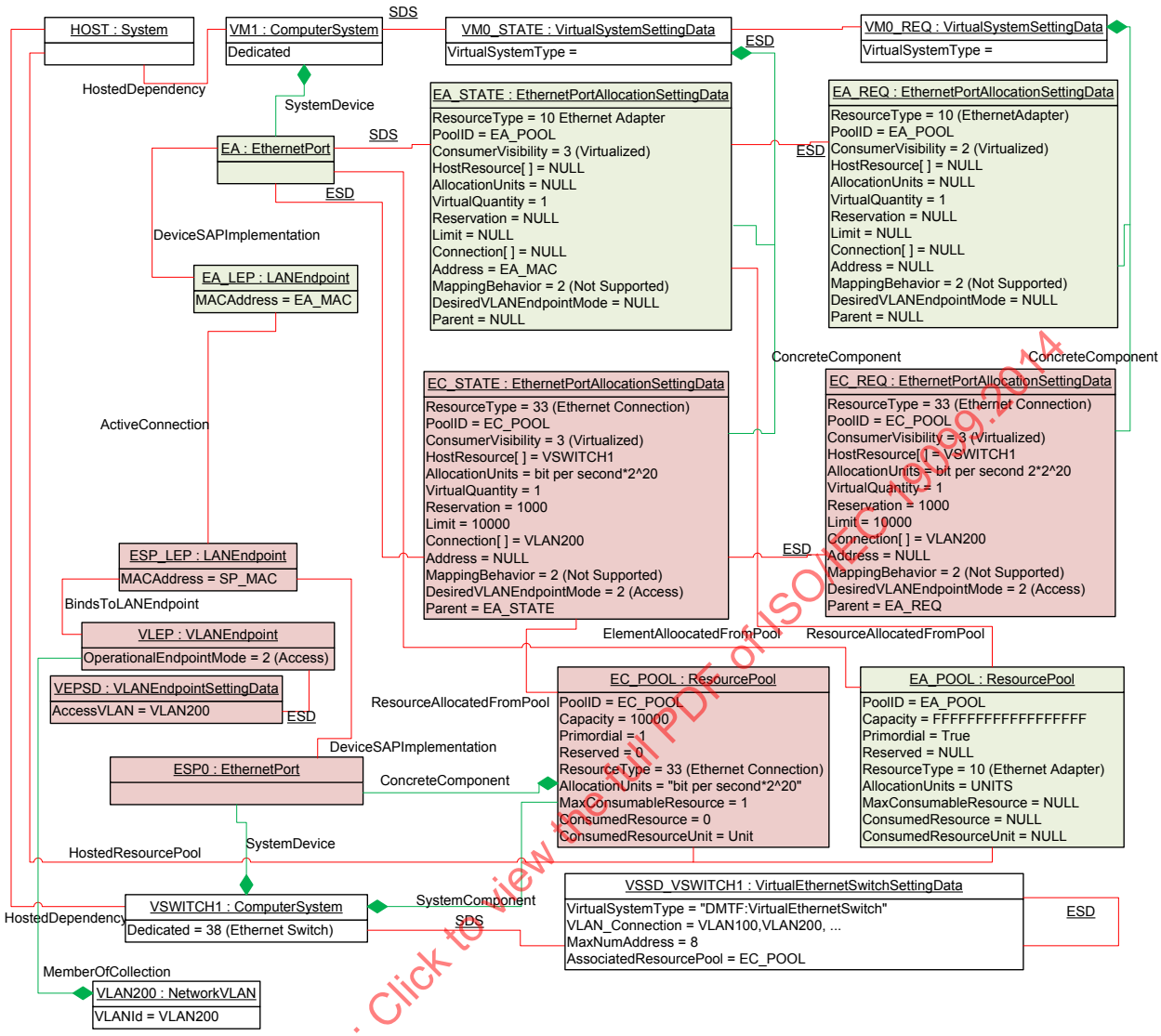

Figure 48 – Dynamic Ethernet switch port allocation

11.4.1.5 Ethernet connection of a virtual system to a virtual switch (simple switch port allocation)

Figure 49 and Figure 50 are CIM instance diagrams that represent a virtualization system that supports a simple connection of a VM to an Ethernet network. Both an implied Ethernet adapter and an Ethernet switch port CIM_EthernetPort instance are instantiated as a result of an Ethernet connection allocation.

Figure 49 is an instance diagram of the allocation capabilities (CAP_EC) of an Ethernet connection resource pool (EC_POOL) associated with a virtual Ethernet switch (VSWITCH2).

The resource pool EC_POOL is identical to the pool shown in Figure 47 and described in 11.4.1.4. The set of capabilities also closely matches the capabilities shown in Figure 47 and described in 11.4.1.4, but the one defining difference is that no valid value (NULL) for the Parent property is shown. Thus, a valid Ethernet connection request can be made without requiring the value of an existing Ethernet adapter request reference to be set in the Parent property.

As a side effect of an Ethernet connection allocation in response to the Ethernet connection request instance EC_REQ, an Ethernet adapter (EA) and an Ethernet switch port (ESP0) are instantiated. EA is associated to VM1 using the CIM_SystemDevice association. ESP0 is associated to the VSWITCH2 using the CIM_SystemDevice association. An instance of CIM_LANEndpoint is instantiated for each of the CIM_EthernetPort instances and associated through the CIM_ActiveConnection association. Also, an Instance of CIM_VLANEndpoint and CIM_VLANEndpointSettingData are instantiated with their properties populated as described in 11.4.1.4.

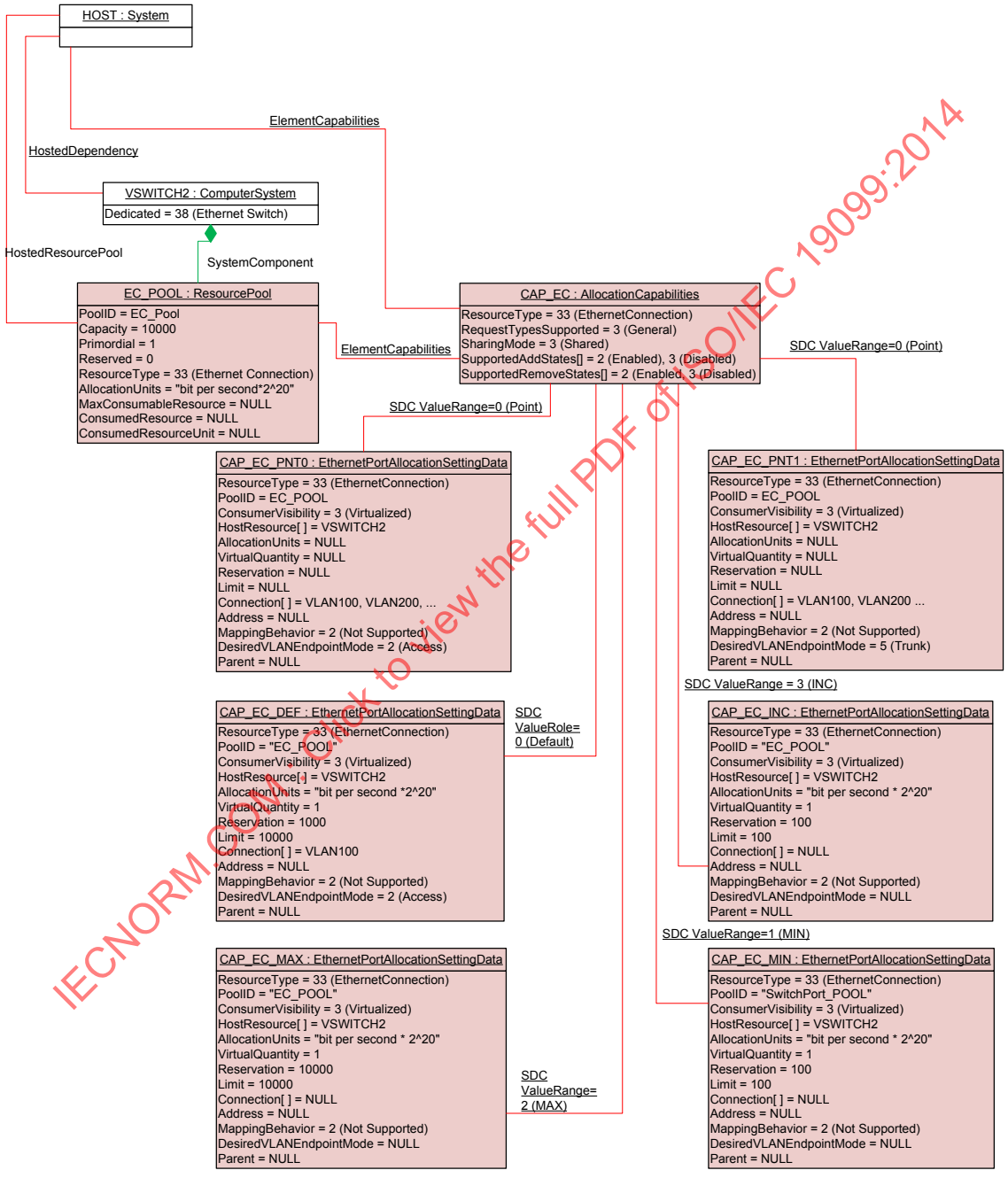

Figure 49 – Allocation capabilities for simple Ethernet connection

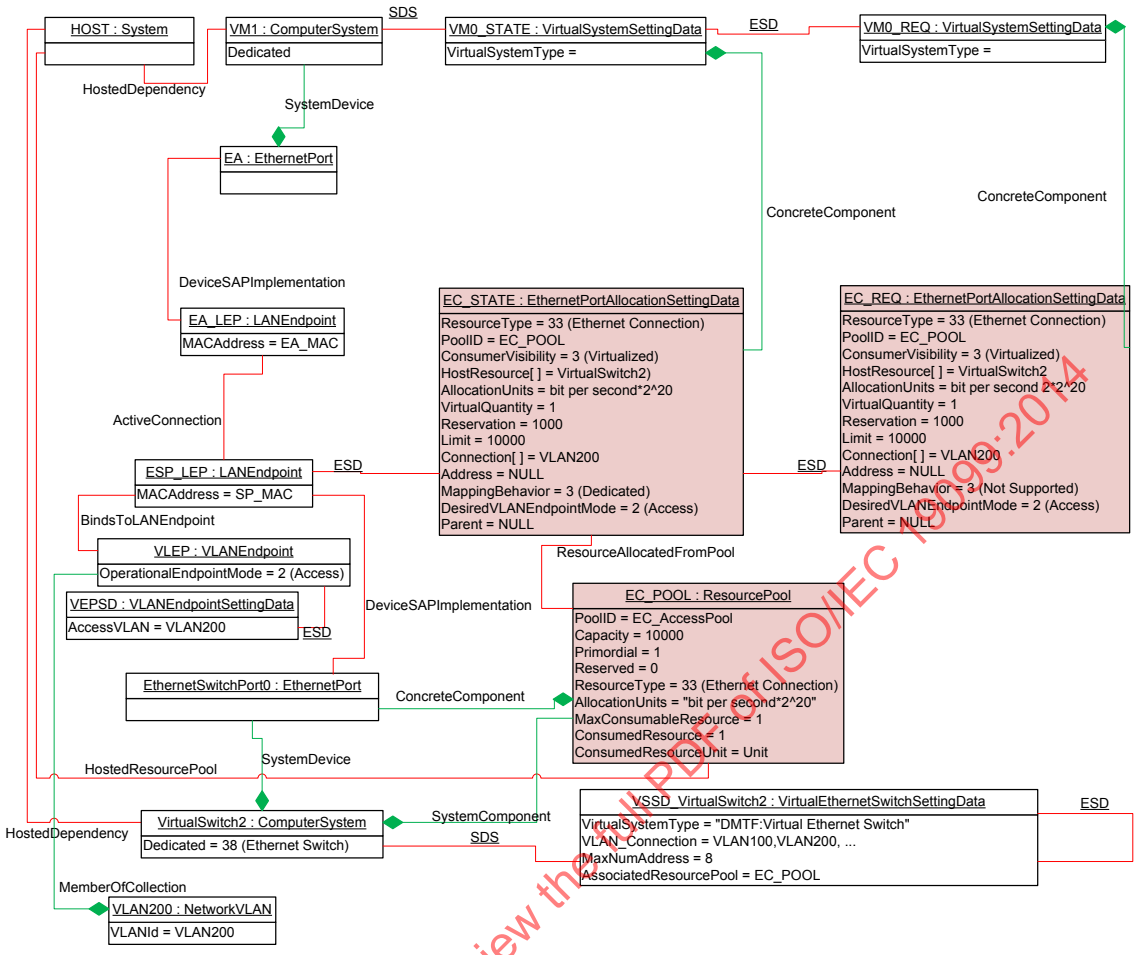

Figure 50 – Simple connection of virtual machine to Ethernet switch

11.4.2 Management

This set of use cases describes how to connect a virtual system to a virtual Ethernet switch. These management tasks are described in terms of a virtual system management service, as represented by a CIM_VirtualSystemManagementService instance.

11.4.2.1 Connection of an Ethernet adapter to a static Ethernet switch port

11.4.2.1.1 Preconditions

All of the following:

- The client knows a reference to the CIM_ComputerSystem instance that represents the virtual system.
- The client knows a reference to the CIM_VirtualSystemManagementService instance that represents the virtual system management service responsible for the virtual system.
- The client has performed the use case and knows the default allocation capabilities of the system.

- The client knows a reference to an available Ethernet switch port on the target virtual Ethernet switch.
- The client knows a reference to an Ethernet adapter request on the target virtual system.

11.4.2.1.2 Flow of activities

1) The client locally prepares an EASD instance, with properties set as follows:

–	ResourceType:	33 (Ethernet Connection)	// Device type as seen by // consumer
–	ResourceSubtype:	NULL	// Implementation dependent
–	PoolID:	NULL	// Implies default pool
–	AllocationUnits:	"bits per second 2*2^20"	/// Units are in megabit per second // bandwidth
–	Reservation:	1000	// 1 gigabit per second bandwidth
–	VirtualQuantityUnits:	"count" // NULL, the effective value	// Count of blocks; if value is // is implied by pool capabilities
–	VirtualQuantity:	1	// One connection
–	Limit:	NULL	// Defaults to maximum limit

–	Address:	NULL	// Optional; if not specified, the
			// implementation uses the current
			// MAC address of the targeted
			// switch port

–	MappingBehavior:	3 (Dedicated)	// Selecting a specific switch port // for exclusive use
–	Parent:	REF to the EASD instance that represents the “defined” targeted Ethernet adapter configuration	
–	HostResource[ ]:	REF to the EASD instance that represents the “defined” targeted Ethernet switch port configuration	
–	DesiredVLANEndpointMode:	2 (Access)	// Set virtual Ethernet switch port // to Access mode.
–	Connection:	VLAN200	// Desired Access VLANID

– Values of all other properties are not set (NULL), requesting a default behavior

2) The client invokes the AddResourceSettings( ) method of the virtual system management service, with parameters set as follows:

–	AffectedConfiguration:	REF to the VSSD instance that represents the "defined" virtual system configuration
–	ResourceSettings:	One element with the embedded EASD instance prepared in step 1)

3) The implementation executes the AddResourceSettings( ) method.

– It is assumed that the method returns 0, indicating successful synchronous execution.

11.4.2.1.3 Postconditions

The virtual Ethernet adapter is connected to the virtual Ethernet switch port, as requested (see Figure 45).

11.4.2.2 Connection of an Ethernet adapter to a dynamic Ethernet switch port

11.4.2.2.1 Preconditions

All of the following:

- The client knows a reference to the CIM_ComputerSystem instance that represents the virtual system.
- The client knows a reference to the CIM_VirtualSystemManagementService instance that represents the virtual system management service responsible for the virtual system.
- The client has performed the use case and knows the default allocation capabilities of the system.
- The client knows a reference to the target virtual Ethernet switch.
- The client knows a reference to an Ethernet adapter request on the target virtual system.

11.4.2.2.2 Flow of activities

The client locally prepares an EASD instance, with properties as specified in use case 11.4.1.4 with the following change:

HostResource[ ]: REF to the CIM_VirtualEthernetSwitchSettingData representing the "defined" configuration of the targeted virtual Ethernet switch

11.4.2.2.3 Postconditions

The implementation creates an instance of CIM_EthernetPort and the required associated protocol endpoints representing an Ethernet switch port and connects the targeted Ethernet adapter to this Ethernet switch port (see Figure 48).

11.5 CIM elements

Table 104 lists CIM elements that are defined or specialized for the profile described in this clause. Each CIM element shall be implemented as described in Table 104. The CIM Schema descriptions for any referenced element and its sub-elements apply.

Subclauses 11.2 ("Implementation") and 11.3 ("Methods") may impose additional requirements on these elements.

Table 171 – CIM Elements: Ethernet Port Resource Virtualization Profile

Element	Requirement	Description
Classes		
CIM_ActiveConnection	Mandatory	See 11.5.1.
CIM_AllocationCapabilities for capabilities	Mandatory	See the Allocation Capabilities Profile described in clause 7.

Element	Requirement	Description
CIM_AllocationCapabilities for mutability	Optional	See the Allocation Capabilities Profile described in clause 7.
CIM_Component for resource pool	Optional	See 11.5.2.
CIM_ElementAllocatedFromPool	Mandatory	See 11.5.3.
CIM_ElementSettingData for Ethernet port resource	Mandatory	See 11.5.4.
CIM_ElementSettingData Ethernet port resource allocation request	Mandatory	See 11.5.5.
CIM_ElementCapabilities for capabilities	Mandatory	See the Allocation Capabilities Profile described in clause 7.
CIM_ElementCapabilities for mutability	Conditional	See the Allocation Capabilities Profile described in clause 7.
CIM_ElementCapabilities for resource pool	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_ElementSettingData for connection resources	Mandatory	See 11.5.4.
CIM_ElementSettingData for CIM_EthernetPort resource allocation	Conditional	See 11.5.5.
CIM_ELementSettingData for CIM _VLANEndpointSetttingData	Conditional	See 11.5.6
CIM_EthernetPort for host systems	Conditional	See 11.5.7.
CIM_EthernetPort for virtual systems	Mandatory	See 11.5.8.
CIM_EthernetPortAllocationSettingData for Ethernet Adapter (Q_EASD)	Optional	See 11.5.9.
CIM_EthernetPortAllocationSettingData for Ethernet Adapter (R_EASD)	Optional	See 11.5.10.
CIM_EthernetPortAllocationSettingData for Ethernet Adapter (C_EASD)	Optional	See 11.5.11.
CIM_EthernetPortAllocationSettingData for Ethernet Adapter (D_EASD)	Optional	See 11.5.12.
CIM_EthernetPortAllocationSettingData for Ethernet Adapter (M_EASD)	Optional	See 11.5.13.
CIM_EthernetPortAllocationSettingData for Ethernet Connection (Q_EASD)	Mandatory	See 11.5.14.
CIM_EthernetPortAllocationSettingData for Ethernet Connection (R_EASD)	Mandatory	See 11.5.15.
CIM_EthernetPortAllocationSettingData for Ethernet Connection (C_EASD)	Mandatory	See 11.5.16.
CIM_EthernetPortAllocationSettingData for Ethernet Connection (D_EASD)	Mandatory	See 11.5.17.
CIM_EthernetPortAllocationSettingData for Ethernet Connection (M_EASD)	Mandatory	See 11.5.18.
CIM_EthernetPortAllocationSettingData for Ethernet Switch Port (Q_EASD)	Optional	See 11.5.19.

Element	Requirement	Description
CIM_EthernetPortAllocationSettingData for Ethernet Switch Port (R_EASD)	Optional	See 11.5.20.
CIM_EthernetPortAllocationSettingData for Ethernet Switch Port (C_EASD)	Optional	See 11.5.21.
CIM_EthernetPortAllocationSettingData for Ethernet Switch Port (D_EASD)	Optional	See 11.5.22.
CIM_EthernetPortAllocationSettingData for Ethernet Switch Port (M_EASD)	Optional	See 11.5.23.
CIM_ReferencedProfile	Mandatory	See 11.5.20.
CIM_RegisteredProfile	Mandatory	See 11.5.24.
CIM_ResourceAllocatedFromPool	Mandatory	See the Resource Allocation Profile described in clause 5.
CIM_ResourcePool Ethernet Adapter	Optional	See 11.5.25.
CIM_ResourcePool Ethernet Connection	Mandatory	See 11.5.26.
CIM_ResourcePool Ethernet Switch Port	Optional	See 11.5.27.
CIM_SettingsDefineState	Mandatory	See 8.5.13.
CIM_SystemDevice (Virtual EthernetPort)	Mandatory	See 11.5.29.
CIM_SystemDevice (Host EthernetPort)	Optional	See 11.5.30.
CIM_VLANEndpointSettingData	Optional	See 11.5.31.

11.5.1 CIM_ActiveConnection

An instance of the CIM_Connection association that associates two instances of the CIM_LANEndPoint class that represents an Ethernet connection between the two CIM_LANEndpoint instances.

Table 172 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema.

Table 172 – Association: CIM_ActiveConnection

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Value shall reference an instance of the CIM_LANEndpoint of an EthernetPort. **Cardinality:** 0..1
Dependent	Mandatory	Key: Value shall reference an instance of the CIM_LANEndpoint of an EthernetPort. **Cardinality:** 0..1
IsUnidirectional	Mandatory	False

11.5.2 CIM_Component for resource pool

The implementation of the CIM_Component association for the representation of the aggregation of host resources into resource pools is conditional.

Condition: The resource aggregation feature (see 11.2.5.10) is implemented.

The CIM_Component association is abstract; therefore, it cannot be directly implemented. For this reason, the provisions in this subclause shall be applied to implementations of subclasses of the CIM_Component association. However, note that clients may directly resolve abstract associations without knowledge of the concrete subclass that is implemented.

Table 173 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 173 – Association: CIM_Component for resource pool

Elements	Requirement	Notes
GroupComponent	Mandatory	**Key:** Value shall reference the instance of the CIM_ResourcePool class that represents an EthernetPort resource pool. **Cardinality:** 0..1
PartComponent	Mandatory	Key: Value shall reference an instance of the CIM_EthernetPort class that represents an Ethernet adapter or Ethernet switch port aggregated into the pool. **Cardinality:** *

11.5.3 CIM_ElementAllocatedFromPool

Table 174 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 174 – Association: CIM_ElementAllocatedFromPool

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Value shall reference the instance of the CIM_ResourcePool class that represents an Ethernet adapter or Ethernet switch port resource pool. **Cardinality:** 1
Dependent	Mandatory	**Key:** Value shall reference the instance of the CIM_EthernetPort class that represents a virtual EthernetPort resulting from an Ethernet adapter or Ethernet switch port allocation from the pool. **Cardinality:** *

11.5.4 CIM_ElementSettingData for connection resources

The CIM_ElementSettingData association associates an instance of the CIM_EthernetPortAllocationSettingData class that represents an Ethernet connection resource allocation and the instance of the CIM_LANEndPoint class associated to the CIM_EthernetPort that represents the targeted Ethernet adapter.

Table 175 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 175 – Association: CIM_ElementSettingData for connection resources

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Value shall reference the instance of the CIM_LANEndpoint class that represents an associated CIMLANEndpoint of the target Ethernet adapter for a connection resource allocation. **Cardinality:** 1
SettingData	Mandatory	**Key:** Value shall reference the instance of the CIM_EthernetPortAllocationSettingData class that represents a corresponding resource allocation request. **Cardinality:** 0..1

11.5.5 CIM_ElementSettingData for CIM_EthernetPort resource allocation

The use of the CIM_ElementSettingData association that is used to associate an instance of CIM_EthernetPortAllocationSettingData representing the allocation of an EthernetPort with a corresponding instance of CIM_EthernetPortAllocationSettingData that describes the same allocation for use as an allocation definition (see the Resource Allocation Profile in clause 5) is conditional.

Condition: The support of the allocation of virtual Ethernet adapters or of virtual Ethernet switch ports.

Table 176 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 176 – Association: CIM_ElementSettingData for CIM_EthernetPort resource allocation

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Value shall reference the instance of the CIM_EthernetPortAllocationSettingData class that represents an Ethernet Adapter or Ethernet switch port resource allocation. **Cardinality:** 1
SettingData	Mandatory	**Key:** Value shall reference the instance of the CIM_EthernetPortAllocationSettingData class that represents a corresponding resource allocation request. **Cardinality:** 0..1

11.5.6 CIM_ElementSettingData for CIM_VLANEndpointSettingData

This use of CIM_ElementSettingData is used to associate a VLAN endpoint's configuration data with an instance of CIM_VLANEndpoint.

Condition: The support for this use of the CIM_ElementSettingData is required if VLAN is supported for an Ethernet port's protocol endpoint.

Table 177 lists the requirements for elements of this class.

Table 177 – Association: CIM_ElementSettingData for CIM_EthernetPort resource allocation

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Value shall reference the instance of the CIM_VLANEndpoint class that represents a VLAN protocol endpoint. **Cardinality:** 1
SettingData	Mandatory	**Key:** Value shall reference the instance of the CIM_VLANEndpointSettingData that represents the configuration data for the VLAN endpoint. **Cardinality:** 0..1

11.5.7 CIM_EthernetPort (host system)

The implementation of the CIM_EthernetPort class for the representation of host Ethernet adapter is conditional.

Condition: The support is required if the CIM_SystemDevice association is supported for the representation of a host Ethernet adapter or a host switch port; see 11.2.3. Table 178 lists the requirements for elements of this class.

Table 178 – Class: CIM_EthernetPort (host system)

Elements	Requirement	Notes
SystemCreationClassName	Mandatory	**Key**
CreationClassName	Mandatory	**Key**
SystemName	Mandatory	**Key**
Name	Mandatory	**Key**

11.5.8 CIM_EthernetPort (virtual system)

See 11.2.7.1 for detailed implementation requirements for this class if it is used for the representation of a virtual Ethernet adapter or an Ethernet switch port.

Table 179 lists the requirements for elements of this class.

Table 179 – Class: CIM_EthernetPort (virtual system)

Elements	Requirement	Notes
SystemCreationClassName	Mandatory	**Key**
CreationClassName	Mandatory	**Key**
SystemName	Mandatory	**Key**
Name	Mandatory	**Key**

11.5.9 CIM_EthernetPortAllocationSettingData for Ethernet adapter (Q_EASD)

See 11.2.6 for detailed implementation requirements for this class.

Table 180 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 180 – Class: CIM_EthernetPortAllocationSettingData for Ethernet adapter (Q_EASD)

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**; see 11.2.6.2.1.13.
ResourceType	Mandatory	Value shall be 10 (Ethernet Adapter). See 11.2.6.2.1.1.
ResourceSubType	Optional	See 11.2.6.2.1.2.
OtherResourceType	Mandatory	Value shall be NULL.
PoolID	Mandatory	See 11.2.6.2.1.3.
ConsumerVisibility	Optional	See 11.2.6.2.1.4.
HostResource[ ]	Optional	See 11.2.6.2.1.6.
AllocationUnits	Mandatory	See 11.2.6.2.1.5.
VirtualQuantity	Mandatory	See 11.2.6.2.1.7.
Reservation	Optional	See 11.2.6.2.1.9.
VirtualQuantityUnits	Mandatory	See 11.2.6.2.1.8.
Limit	Optional	See 11.2.6.2.1.10.
Weight	Optional	See 11.2.6.2.1.11.
Parent	Optional	See 11.2.6.2.1.12.
Address	Optional	See 11.2.6.2.1.13.
Connection[ ]	Optional	See 11.2.6.2.1.15.
MappingBehavior	Optional	See 11.2.6.2.1.16.
AutomaticAllocation	Optional	See the Resource Allocation Profile described in clause 5.
AutomaticDeallocation	Optional	See the Resource Allocation Profile described in clause 5.

11.5.10 CIM_EthernetPortAllocationSettingData for Ethernet adapter (R_EASD)

See 11.2.6 for detailed implementation requirements for this class.

Table 181 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 181 – Class: CIM_EthernetPortAllocationSettingData for Ethernet adapter (R_EASD)

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**; see 11.2.6.2.1.13.
ResourceType	Mandatory	Value shall be 10 (Ethernet Adapter). See 11.2.6.2.1.1.

Elements	Requirement	Notes
ResourceSubType	Optional	See 11.2.6.2.1.2.
OtherResourceType	Mandatory	Value shall be NULL.
PoolID	Mandatory	See 11.2.6.2.1.3.
ConsumerVisibility	Optional	See 11.2.6.2.1.4.
HostResource[ ]	Optional	See 11.2.6.2.1.6.
AllocationUnits	Mandatory	See 11.2.6.2.1.5.
VirtualQuantity	Mandatory	See 11.2.6.2.1.7.
Reservation	Optional	See 11.2.6.2.1.9.
VirtualQuantityUnits	Mandatory	See 11.2.6.2.1.8.
Limit	Optional	See 11.2.6.2.1.10.
Weight	Optional	See 11.2.6.2.1.11.
Address	Mandatory	See 11.2.6.2.1.13.
Parent	Optional	See 11.2.6.2.1.12.
Connection[ ]	Optional	See 11.2.6.2.1.15.
MappingBehavior	Optional	See 11.2.6.2.1.16.
AutomaticAllocation	Optional	See the Resource Allocation Profile described in clause 5.
AutomaticDeallocation	Optional	See the Resource Allocation Profile described in clause 5.

11.5.11 CIM_EthernetPortAllocationSettingData for Ethernet adapter (C_EASD)

See 11.2.6 for detailed implementation requirements for this class.

Table 182 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 182 – Class: CIM_EthernetPortAllocationSettingData for Ethernet adapter (C_EASD)

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**; see 11.2.6.2.1.13.
ResourceType	Mandatory	Value shall be 10 (Ethernet Adapter). See 11.2.6.2.1.1.
ResourceSubType	Optional	See 11.2.6.2.1.2.
OtherResourceType	Mandatory	Value shall be NULL.
PoolID	Mandatory	See 11.2.6.2.1.3.
ConsumerVisibility	Optional	See 11.2.6.2.1.4.
HostResource[ ]	Optional	See 11.2.6.2.1.6.
AllocationUnits	Mandatory	See 11.2.6.2.1.5.
VirtualQuantity	Mandatory	See 11.2.6.2.1.7.
Reservation	Optional	See 11.2.6.2.1.9.

Elements	Requirement	Notes
VirtualQuantityUnits	Mandatory	See 11.2.6.2.1.8.
Limit	Optional	See 11.2.6.2.1.10.
Weight	Optional	See 11.2.6.2.1.11.
Address	Optional	See 11.2.6.2.1.13.
Parent	Optional	See 11.2.6.2.1.12.
Connection[ ]	Optional	See 11.2.6.2.1.15.
MappingBehavior	Optional	See 11.2.6.2.1.16.
AutomaticAllocation	Optional	See the Resource Allocation Profile described in clause 5.
AutomaticDeallocation	Optional	See the Resource Allocation Profile described in clause 5.

11.5.12 CIM_EthernetPortAllocationSettingData for Ethernet adapter (D_EASD)

See 11.2.6 for detailed implementation requirements for this class.

Table 183 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 183 – Class: CIM_EthernetPortAllocationSettingData for Ethernet adapter (D_EASD)

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**; see 11.2.6.2.1.13.
ResourceType	Mandatory	Value shall be 10 (Ethernet Adapter). See 11.2.6.2.1.1.
ResourceSubType	Optional	See 11.2.6.2.1.2.
OtherResourceType	Mandatory	Value shall be NULL.
PoolID	Mandatory	See 11.2.6.2.1.3.
ConsumerVisibility	Optional	See 11.2.6.2.1.4.
HostResource[ ]	Optional	See 11.2.6.2.1.6.
AllocationUnits	Mandatory	See 11.2.6.2.1.5.
VirtualQuantity	Mandatory	See 11.2.6.2.1.7.
Reservation	Optional	See 11.2.6.2.1.9.
VirtualQuantityUnits	Mandatory	See 11.2.6.2.1.8.
Limit	Optional	See 11.2.6.2.1.10.
Weight	Optional	See 11.2.6.2.1.11.
Address	Optional	See 11.2.6.2.1.13.
Parent	Optional	See 11.2.6.2.1.12.
Connection[ ]	Optional	See 11.2.6.2.1.15.
MappingBehavior	Optional	See 11.2.6.2.1.16.

Elements	Requirement	Notes
AutomaticAllocation	Optional	See the Resource Allocation Profile described in clause 5.
AutomaticDeallocation	Optional	See the Resource Allocation Profile described in clause 5.

11.5.13 CIM_EthernetPortAllocationSettingData for Ethernet adapter (M_EASD)

See 11.2.6 for detailed implementation requirements for this class.

Table 184 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in the Resource Allocation Profile described in clause 5.

Table 184 – Class: CIM_EthernetPortAllocationSettingData for Ethernet adapter (M_EASD)

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**; see 11.2.6.2.1.13.
ResourceType	Mandatory	Value shall be 10 (Ethernet Adapter). See 11.2.6.2.1.1.
ResourceSubType	Optional	See 11.2.6.2.1.2.
OtherResourceType	Mandatory	Value shall be NULL.
PoolID	Mandatory	See 11.2.6.2.1.3.
ConsumerVisibility	Optional	See 11.2.6.2.1.4.
HostResource[ ]	Optional	See 11.2.6.2.1.6.
AllocationUnits	Mandatory	See 11.2.6.2.1.5.
VirtualQuantity	Mandatory	See 11.2.6.2.1.7.
Reservation	Optional	See 11.2.6.2.1.9.
VirtualQuantityUnits	Mandatory	See 11.2.6.2.1.8.
Limit	Optional	See 11.2.6.2.1.10.
Weight	Optional	See 11.2.6.2.1.11.
Address	Optional	See 11.2.6.2.1.13.
Parent	Optional	See 11.2.6.2.1.12.
Connection[ ]	Optional	See 11.2.6.2.1.15.
MappingBehavior	Optional	See 11.2.6.2.1.16.
AutomaticAllocation	Optional	See the Resource Allocation Profile described in clause 5.
AutomaticDeallocation	Optional	See the Resource Allocation Profile described in clause 5.

11.5.14 CIM_EthernetPortAllocationSettingData for Ethernet connection (Q_EASD)

See 11.2.6 for detailed implementation requirements for this class.

Table 185 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema.

Table 185 – Class: CIM_EthernetPortAllocationSettingData for Ethernet connection (Q_EASD)

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
ResourceType	Mandatory	Value shall be 33 (Ethernet Connection). See 11.2.6.2.1.1.
ResourceSubType	Optional	See 11.2.6.2.1.2.
OtherResourceType	Mandatory	Value shall be NULL.
PoolID	Mandatory	See 11.2.6.2.1.3.
ConsumerVisibility	Optional	See 11.2.6.2.1.4.
HostResource[ ]	Optional	See 11.2.6.2.1.6.
AllocationUnits	Mandatory	See 11.2.6.2.1.5.
VirtualQuantity	Mandatory	See 11.2.6.2.1.7.
Reservation	Optional	See 11.2.6.2.1.9.
VirtualQuantityUnits	Mandatory	See 11.2.6.2.1.8.
Limit	Optional	See 11.2.6.2.1.10.
Weight	Optional	See 11.2.6.2.1.11.
Address	Optional	See 11.2.6.2.1.13.
Parent	Optional	See 11.2.6.2.1.12.
Connection[ ]	Optional	See 11.2.6.2.1.15.
MappingBehavior	Optional	See 11.2.6.2.1.16.
AutomaticAllocation	Optional	See the Resource Allocation Profile described in clause 5.
AutomaticDeallocation	Optional	See the Resource Allocation Profile described in clause 5.

11.5.15 CIM_EthernetPortAllocationSettingData for Ethernet connection (R_EASD)

See 11.2.6 for detailed implementation requirements for this class.

Table 186 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema.

Table 186 – Class: CIM_EthernetPortAllocationSettingData for Ethernet connection (R_EASD)

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
ResourceType	Mandatory	Value shall be 33 (Ethernet Connection). See 11.2.6.2.1.1.
ResourceSubType	Optional	See 11.2.6.2.1.2.
OtherResourceType	Mandatory	Value shall be NULL.
PoolID	Mandatory	See 11.2.6.2.1.3.
ConsumerVisibility	Optional	See 11.2.6.2.1.4.
HostResource[ ]	Optional	See 11.2.6.2.1.6.
AllocationUnits	Mandatory	See 11.2.6.2.1.5.
VirtualQuantity	Mandatory	See 11.2.6.2.1.7.
Reservation	Optional	See 11.2.6.2.1.9.
VirtualQuantityUnits	Mandatory	See 11.2.6.2.1.8.
Limit	Optional	See 11.2.6.2.1.10.
Weight	Optional	See 11.2.6.2.1.11.
Address	Mandatory	See 11.2.6.2.1.13.
Parent	Optional	See 11.2.6.2.1.12.
Connection[ ]	Optional	See 11.2.6.2.1.15.
MappingBehavior	Optional	See 11.2.6.2.1.16.
AutomaticAllocation	Optional	See the Resource Allocation Profile described in clause 5.
AutomaticDeallocation	Optional	See the Resource Allocation Profile described in clause 5.

11.5.16 CIM_EthernetPortAllocationSettingData for Ethernet connection (C_EASD)

See 11.2.6 for detailed implementation requirements for this class.

Table 187 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema.

Table 187 – Class: CIM_EthernetPortAllocationSettingData for Ethernet connection (C_EASD)

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
ResourceType	Mandatory	Value shall be 33 (Ethernet Connection). See 11.2.6.2.1.1.
ResourceSubType	Optional	See 11.2.6.2.1.2.
OtherResourceType	Mandatory	Value shall be NULL.

Elements	Requirement	Notes
PoolID	Mandatory	See 11.2.6.2.1.3.
ConsumerVisibility	Optional	See 11.2.6.2.1.4.
HostResource[ ]	Optional	See 11.2.6.2.1.6.
AllocationUnits	Mandatory	See 11.2.6.2.1.5.
VirtualQuantity	Mandatory	See 11.2.6.2.1.7.
Reservation	Optional	See 11.2.6.2.1.9.
VirtualQuantityUnits	Mandatory	See 11.2.6.2.1.8.
Limit	Optional	See 11.2.6.2.1.10.
Weight	Optional	See 11.2.6.2.1.11.
Address	Optional	See 11.2.6.2.1.13.
Parent	Optional	See 11.2.6.2.1.12.
Connection[ ]	Optional	See 11.2.6.2.1.15.
MappingBehavior	Optional	See 11.2.6.2.1.16.
AutomaticAllocation	Optional	See the Resource Allocation Profile described in clause 5.
AutomaticDeallocation	Optional	See the Resource Allocation Profile described in clause 5.

11.5.17 CIM_EthernetPortAllocationSettingData for Ethernet connection (D_EASD)

See 11.2.6 for detailed implementation requirements for this class.

Table 188 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema.

Table 188 – Class: CIM_EthernetPortAllocationSettingData for Ethernet connection (D_EASD)

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
ResourceType	Mandatory	Value shall be 33 (Ethernet Connection). See 11.2.6.2.1.1.
ResourceSubType	Optional	See 11.2.6.2.1.2.
OtherResourceType	Mandatory	Value shall be NULL.
PoolID	Mandatory	See 11.2.6.2.1.3.
ConsumerVisibility	Optional	See 11.2.6.2.1.4.
HostResource[ ]	Optional	See 11.2.6.2.1.6.
AllocationUnits	Mandatory	See 11.2.6.2.1.5.
VirtualQuantity	Mandatory	See 11.2.6.2.1.7.
Reservation	Optional	See 11.2.6.2.1.9.
VirtualQuantityUnits	Mandatory	See 11.2.6.2.1.8.
Limit	Optional	See 11.2.6.2.1.10.
Weight	Optional	See 11.2.6.2.1.11.