Document Number: DSP1042

Date: 2009-08-06

Version: 1.0.0e

System Virtualization Profile

Information for Work-in-Progress version:

This document is subject to change at any time without further notice.

It expires on: 2010-03-31

Target version for final status: 1.0.0e

Provide any comments through the DMTF Feedback Portal: http://www.dmtf.org/standards/feedback

Document Type: Specification

Document Status: DMTF Work in Progress - Expires 2010-03-31

Document Language: E

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. Members and non-members may reproduce DMTF specifications and documents, provided that correct attribution is given. As DMTF specifications may be revised from time to time, the particular version and release date should always be noted.

Implementation of certain elements of this standard or proposed standard may be subject to third party patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose, or identify any or all such third party patent right, owners or claimants, nor for any incomplete or inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize, disclose, or identify any such third party patent rights, or for such party's reliance on the standard or incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any party implementing such standard, whether such implementation is foreseeable or not, nor to any patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a patent owner for such implementations.

For information about patents held by third-parties which have notified the DMTF that, in their opinion, such patent may relate to or impact implementations of DMTF standards, visit http://www.dmtf.org/about/policies/disclosures.php.

CONTENTS

183

184 # Figures

192

193 # Tables

Foreword

This profile (DSP1042, System Virtualization Profile) was prepared by the System Virtualization, Partitioning and Clustering Working Group of the DMTF.

The DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability.

Introduction

The information in this specification should be sufficient for a provider or consumer of this data to unambiguously identify the classes, properties, methods, and values that shall be instantiated and manipulated to represent and manage a host system, its resources, and related services, and to create and manipulate virtual systems. The target audience for this specification is implementers who are writing CIM-based providers or consumers of management interfaces that represent the components described in this document.

System Virtualization Profile

1 Scope

This profile 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.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

DMTF DSP0004, *CIM Infrastructure Specification* 2.5
http://www.dmtf.org/standards/published_documents/DSP0004_2.5.pdf

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

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

DMTF 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 DSP1022, *CPU Profile 1.0*
http://www.dmtf.org/standards/published_documents/DSP1022_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 DSP1041, *Resource Allocation Profile 1.1*
http://www.dmtf.org/standards/published_documents/DSP1041_1.1.pdf

DMTF DSP1043, *Allocation Capabilities Profile* 1.0
http://www.dmtf.org/standards/published_documents/DSP1043_1.0.pdf

DMTF DSP1044, *Processor Device Resource Virtualization Profile 1.0*
http://www.dmtf.org/standards/published_documents/DSP1044_1.0.pdf

DMTF DSP1045, *Memory Resource Virtualization Profile 1.0*
http://www.dmtf.org/standards/published_documents/DSP1045_1.0.pdf

DMTF DSP1047, *Storage Resource Virtualization Profile 1.0*
http://www.dmtf.org/standards/published_documents/DSP1047_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 DSP1057, *Virtual System Profile 1.0*
http://www.dmtf.org/standards/published_documents/DSP1057_1.0.pdf

DMTF DSP1059, *Generic Device Resource Virtualization Profile 1.0*
http://www.dmtf.org/standards/published_documents/DSP1059_1.0.pdf

ISO/IEC Directives, Part2:2004, *Rules for the structure and drafting of International Standards*,
http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply. For the purposes of this document, the terms and definitions in DSP1033 and DSP1001 also apply.

3.1

can

used for statements of possibility and capability, whether material, physical, or causal

3.2

cannot

used for statements of possibility and capability, whether material, physical, or causal

3.3

conditional

indicates requirements to be followed strictly in order to conform to the document and from which no deviation is permitted, when the specified conditions are met

3.4

mandatory

indicates requirements to be followed strictly in order to conform to the document and from which no deviation is permitted

3.5

may

indicates a course of action permissible within the limits of the document

3.6

need not

indicates a course of action permissible within the limits of the document

3.7

optional

indicates a course of action permissible within the limits of the document

3.8
referencing profile
indicates a profile that owns the definition of this class and can include a reference to this profile in its "Related Profiles" table

3.9
shall
indicates requirements to be followed strictly in order to conform to the document and from which no deviation is permitted

3.10
shall not
indicates requirements to be followed strictly in order to conform to the document and from which no deviation is permitted

3.11
should
indicates that among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required

3.12
should not
indicates that a certain possibility or course of action is deprecated but not prohibited

3.13
unspecified
indicates that this profile does not define any constraints for the referenced CIM element

3.14
implementation
a set of software components that realize the classes that are specified or specialized by this profile

3.15
client
application that exploits facilities specified by this profile

3.16
this profile
a reference to this DMTF management profile: DSP1042 (System Virtualization Profile)

3.17
virtualization platform
virtualizing infrastructure provided by a host system that enables the deployment of virtual systems

3.18
WBEM service
A component that provides a service accessible through a WBEM protocol. A single WBEM service instance may be used by multiple WBEM client instances. The term WBEM service is used to denote the entire set if components on the server side that is needed to provide the service. For example, in typical WBEM infrastructures this includes a CIM object manager and a set of CIM providers.

4 Symbols and abbreviated terms

The following symbols and abbreviations are used in this document.

4.1

RASD

resource allocation setting data

4.2

SLP

service location protocol

4.3

VS

virtual system

4.4

VSSD

virtual system setting data

5 Synopsis

Profile Name: *System Virtualization*

Version: 1.0.0

Organization: DMTF

CIM Schema Version: 2.22

Central Class: CIM_System

Scoping Class: CIM_System

This 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 DSP1057.

The central instance and the scoping instance of this profile shall be an instance of the CIM_System class that represents a host system.

Table 1 lists DMTF management profiles that this profile depends on, or that may be used in the context of this profile.

Table 1 – Related Profiles

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 7.2.

Profile Name	Organization	Version	Relationship	Description
Virtual System	DMTF	1.0	Mandatory	The autonomous DMTF management profile that specifies the minimum object model needed for the inspection and basic manipulation of a virtual system; see 7.3.
Processor Device Resource Virtualization	DMTF	1.0	Conditional	The component DMTF management profile that specifies the allocation of processor resources; see 7.2.2.
Memory Resource Virtualization	DMTF	1.0	Conditional	The component DMTF management profile that specifies the allocation of memory resources; see 7.2.2.
Storage Adapter Resource Virtualization	DMTF	1.0	Conditional	The component DMTF management profile that specifies the allocation of storage adapter resources; see 7.2.2.
Generic Device Resource Virtualization	DMTF	1.0	Conditional	The component DMTF management profile that specifies the allocation of generic resources; see 7.2.2.

6 Description

This clause contains informative text only.

This profile 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 Profile relationships

A client that is exploiting system virtualization facilities specified by this profile 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.

417 This profile complements DSP1057.

418 • This profile focuses on virtualization aspects related to host systems and their resources, such
419 as modeling the relationships between host resources and virtual resources. Further it
420 addresses virtualization-specific tasks such as the creation or modification of virtual systems
421 and their configurations.

422 • DSP1057 defines a top-level object model for the inspection and basic operation of virtual
423 systems. It is a specialization of DSP1052 that defines a management interface for general-
424 purpose computer systems. Consequently, the interface specified for the basic inspection and
425 operation of virtual systems is conformant with that specified for real systems. A client that is
426 exploiting capabilities specified by DSP1052 with respect to virtual systems that are instrument
427 conformant with DSP1057 can inherently handle virtual systems like real systems without being
428 virtualization aware.

429 Figure 1 shows the structure of DMTF management profiles related to system virtualization.

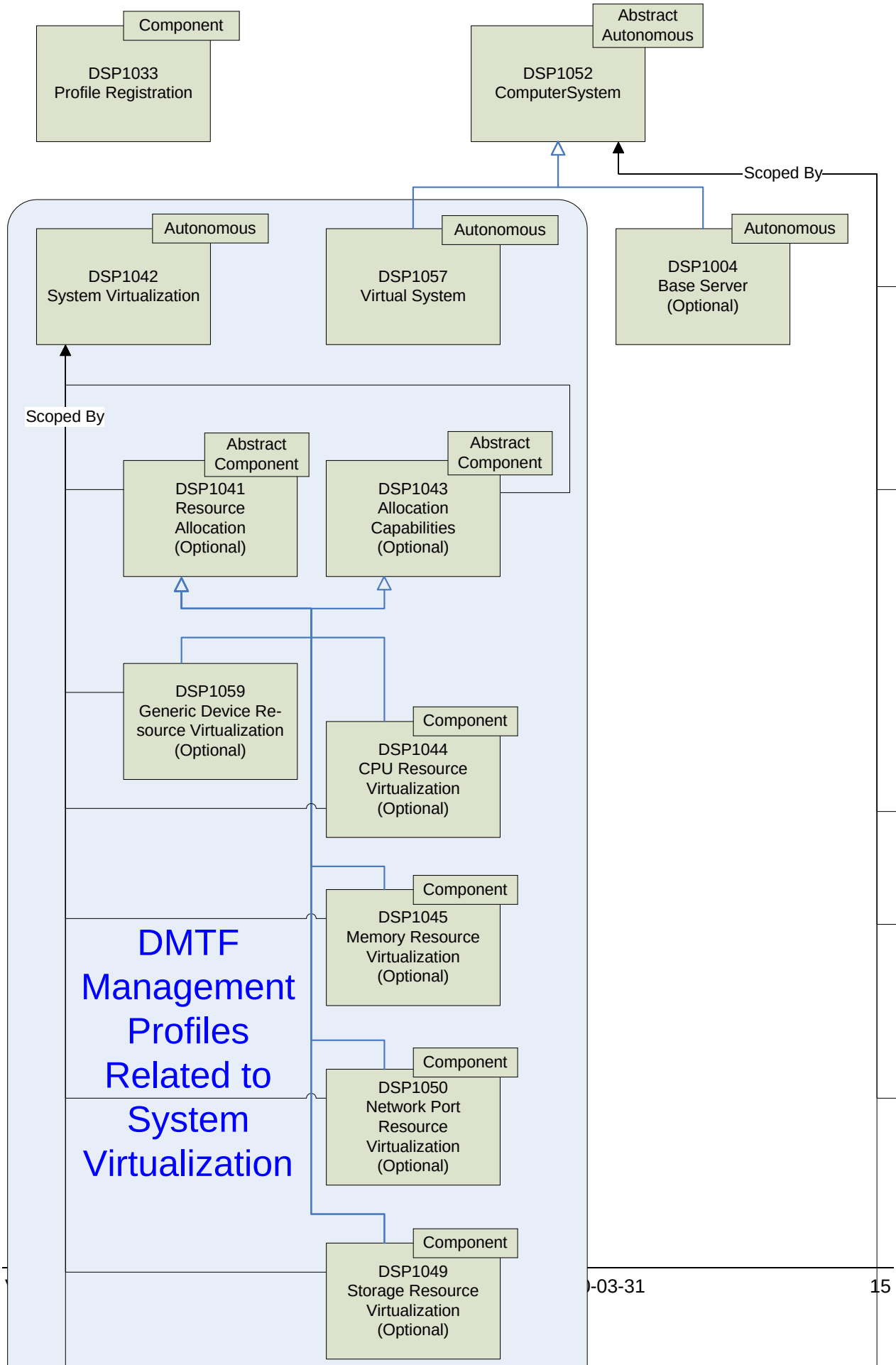
Component
DSP1033
Profile Registration
Abstract
Autonomous
DSP1052
ComputerSystem
Scoped By
Autonomous
DSP1042
System Virtualization
Autonomous
DSP1057
Virtual System
Autonomous
DSP1004
Base Server
(Optional)
Cor
DSP1027
Power Stat
Manageme
(Optional)
Scoped By
Abstract
Component
DSP1041
Resource
Allocation
(Optional)
Abstract
Component
DSP1043
Allocation
Capabilities
(Optional)
Cor
DSP1012
Boot Control P
(Optional)
DSP1059
Generic Device Re-
source Virtualization
(Optional)
Component
DSP1044
CPU Resource
Virtualization
(Optional)
Cor
DSP1022
CPU Profil
(Optional)
Component
DSP1045
Memory Resource
Virtualization
(Optional)
Cor
DSP1026
System Mem
Profile
(Optional)
DMTF
Management
Profiles
Related to
System
Virtualization
Component
DSP1050
Network Port
Resource
Virtualization
(Optional)
Cor
DSP1014
Ethernet Port F
(Optional)
Component
DSP1049
Storage Resource
Virtualization
(Optional)

Figure 1 – Profiles related to system virtualization

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

- This profile

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

- DSP1057

  DSP1057 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 DSP1041 and on DSP1043. Resource allocation profiles are scoped by this profile. 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 DSP1044 and DSP1047.

  The management of the allocation of basic virtual resources that are not covered by a resource-type-specific resource allocation profile is specified in DSP1059.

6.2 System virtualization class schema

Figure 2 shows the complete class schema of this profile. It outlines elements that are specified or specialized by this profile, as well as the dependency relationships between elements of this profile and other profiles. For simplicity in diagrams, the prefix *CIM_* has been removed from class and association names.

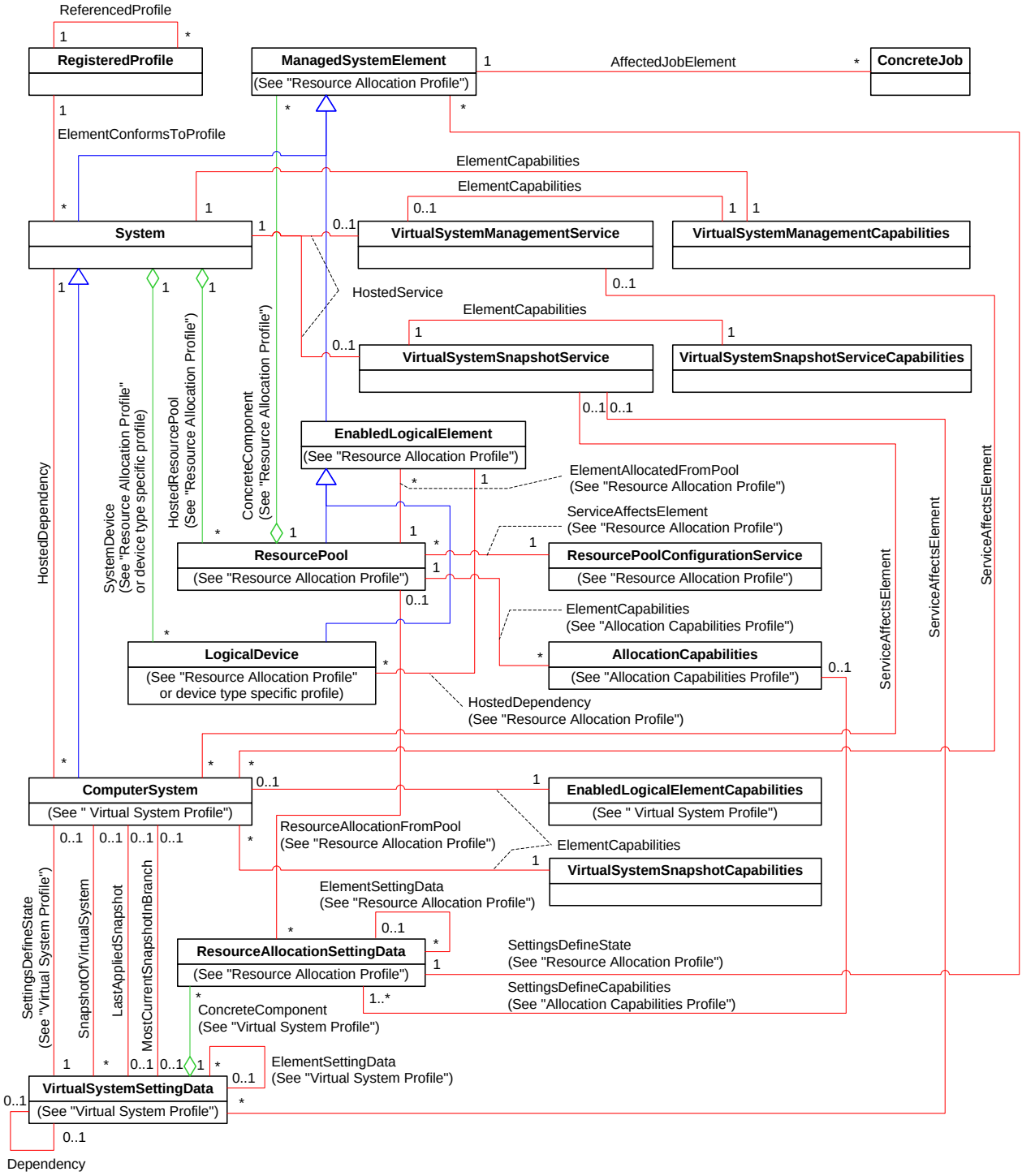

Figure 2 – *System Virtualization Profile*: Class diagram

This 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 this profile 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 DSP1057)
- 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.

6.3 Virtual system configurations

This profile extends the use of virtual system configurations. DSP1057 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.

DSP1057 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

This profile 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

6.4 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. This profile 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.

6.5 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.

This profile 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.

This profile specifies mechanisms that enable the inspection of configuration information of snapshots and their related virtual systems only. This profile 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.

7 Implementation

This clause details the requirements related to classes and their properties for implementations of this profile. The CIM Schema descriptions for any referenced element and its sub-elements apply.

The list of all required methods can be found in 8 ("Methods") and the list of all required properties can be found in 10 ("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.

7.1 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 this profile.

7.2 Profile registration

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

7.2.1 This profile

The implementation of this profile 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 this profile shall be a central instance of this profile by associating it with the instance of the CIM_RegisteredProfile class through an instance of the CIM_ElementConformsToProfile association.

7.2.2 Scoped resource allocation profiles

An implementation of this profile 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 this profile through an instance of the CIM_ReferencedProfile association.

Resource-allocation DMTF management profiles are based on DSP1041 and DSP1043. The resource-allocation DMTF management profiles that are scoped by this profile are listed in Table 1, starting with DSP1044.

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, DSP1059 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 this profile 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.

7.3 Representation of hosted virtual systems

This profile strengthens the requirements for the representation of virtual system configurations specified by DSP1057 for hosted virtual systems.

7.3.1 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 DSP1057. 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.

7.3.2 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 7.4.2).

7.4 Virtual system management capabilities

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

7.4.1 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.

7.4.2 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.

7.4.3 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.

7.4.4 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.

7.4.5 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.

7.4.6 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.

7.4.6.1 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.

7.4.6.2 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.

7.4.6.3 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.

7.5 Virtual system definition and modification

This profile 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.

7.5.1 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 8.2.4.3).

7.5.2 CIM_VirtualSystemSettingData.ElementName property

The implementation of the ElementName property is optional.

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

768 A client may set the value of the ElementName property to assign a user-friendly name to a virtual sys-
769 tem.

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

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

776 7.5.3 CIM_VirtualSystemSettingData.VirtualSystemIdentifier property

777 The implementation of the VirtualSystemIdentifier property is optional.

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

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

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

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

789 7.5.4 CIM_VirtualSystemSettingData.VirtualSystemType property

790 The implementation of the VirtualSystemType property is optional.

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

793 A client may set the value of the VirtualSystemType property to explicitly request a virtual system type for
794 the new virtual system. A client may set the value of the VirtualSystemType property to NULL, requesting
795 the implementation to assign a virtual system type according to rules specified in this subclause. If
796 requesting a value other than NULL, the client should determine the list of valid system types in advance
797 (see 9.2.7).

798 An implementation shall use the value of the VirtualSystemType property to assign a type to the new vir-
799 tual system. If the value of the VirtualSystemType property is NULL, the implementation shall assign a
800 virtual system type in an implementation-dependent way. If the requested virtual system type is not sup-
801 ported, an implementation shall fail the method execution with an error code of 4 (Method execution failed
802 because invalid parameters were specified by the client).

803 7.6 Virtual resource definition and modification

804 This profile specifies how to define and modify virtual resources using methods of the virtual system
805 management service. In these method specifications, the CIM_ResourceAllocationSettingData class is
806 used for parameterization of resource allocation specific properties. For specifications that define the use
807 of the CIM_ResourceAllocationSettingData class, see DSP1041, DSP1043, and profiles that specialize
808 these (such as for example, DSP1059). DSP1041 describes the use of the

809 CIM_ResourceAllocationSettingData class, and DSP1043 introduces the concept of allowing a client to
810 determine the acceptable value sets for values of properties of the CIM_ResourceAllocationSettingData
811 class in virtual resource definition and modification requests.

812 7.7 Virtual system snapshots

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

814 The implementation of virtual system snapshots is optional.

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

816 7.7.1 Virtual system snapshot service and capabilities

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

819 7.7.1.1 Virtual system snapshots

820 The implementation of virtual system snapshots is optional.

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

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

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

824 • the CIM_VirtualSystemSnapshotService class shall be implemented and the following methods
825 shall be implemented:
826 – CreateSnapshot( ), for at least one type of snapshot
827 – DestroySnapshot( )
828 – ApplySnapshot( )
829 • There shall be exactly one instance of the CIM_VirtualSystemSnapshotService class associated
830 to the central instance of this profile through an instance of the CIM_HostedService association.

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

833 7.7.1.2 CIM_VirtualSystemSnapshotServiceCapabilities class

834 The provisions in this subclause are conditional.

835 Condition: Virtual system snapshots are implemented; see 7.7.1.1.

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

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

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

845 • 2 (CreateSnapshotSupported)
846 • 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 aynchronous 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.

7.7.2 Virtual system snapshot representation

The provisions in this subclause are conditional.

Condition: Virtual system snapshots are implemented; see 7.7.1.1.

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.

7.7.3 Designation of the last applied snapshot

The provisions in this subclause are conditional.

Condition: Virtual system snapshots are implemented; see 7.7.1.1.

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.

7.7.4 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 7.7.1.1.

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.

7.7.5 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.

7.7.5.1 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.

7.7.5.2 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.

8 Methods

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

8.1 General behavior of extrinsic methods

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

8.1.1 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 DSP1041.

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.

8.1.2 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).

8.1.3 Asynchronous processing

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

8.1.3.1 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 7.4.4.

966 • Elements that convey information about which methods of the
967 CIM_VirtualSystemSnapshotService class are implemented for asynchronous execution within
968 an implementation are modeled in 7.7.1.1.

969 • Elements that convey information about whether a method is executed asynchronously are
970 modeled in 8.1.2.

971 #### 8.1.3.2 Job parameter

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

974 • If the method execution is performed synchronously, the implementation shall set the value to
975 NULL.

976 • If the method execution is performed asynchronously, the implementation shall set the value to
977 refer to the instance of the CIM_ConcreteJob class that represents the asynchronous task.

978 ## 8.2 Methods of the CIM_VirtualSystemManagementService class

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

981 ### 8.2.1 CIM_VirtualSystemManagementService.DefineSystem( ) method

982 The implementation of the DefineSystem( ) method is conditional.

983 Condition: The definition and destruction of virtual systems is implemented; see 7.4.6.1.

984 If the DefineSystem( ) method is implemented, the provisions in this subclause apply; in addition behavior
985 applicable to all extrinsic methods is specified in 8.1.2.

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

991 Table 2 contains requirements for parameters of this method.

992 **Table 2 – DefineSystem( ) method: Parameters**

Qualifiers	Name	Type	Description/Values
IN	SystemSettings	string	See 8.2.1.2.
IN	ResourceSettings[ ]	string	See 8.2.1.3.
IN	ReferencedConfiguration	CIM_VirtualSystemSettingData REF	See 8.2.1.4.
OUT	ResultingSystem	CIM_ComputerSystem REF	See 8.2.1.5.
OUT	Job	CIM_ConcreteJob REF	See 8.1.3.2.

8.2.1.1 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. DSP1059 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 in DSP1057).

8.2.1.2 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 8.2.1.1.

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 8.2.1.1.

The use of the CIM_VirtualSystemSettingData class as input for operations specified by this profile is specified in 10.22.

8.2.1.3 ResourceSettings[ ] array parameter

A client should set the ResourceSettings[ ] array parameter and apply the specifications given in 8.1.1. 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 8.2.1.1).

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 8.2.1.1.

8.2.1.4 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 8.2.1.1.

8.2.1.5 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.

8.2.1.6 Return codes

An implementation shall indicate the result of the method execution by using the return code values specified in Table 3.

Table 3 – 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 8.1.3 apply.

8.2.2 CIM_VirtualSystemManagementService.DestroySystem( ) method

The implementation of the DestroySystem( ) method is conditional.

Condition: The definition and destruction of virtual systems is implemented; see 7.4.6.1.

If the DestroySystem( ) method is implemented, the provisions in this subclause apply; in addition behavior applicable to all extrinsic methods is specified in 8.1.2.

1067 The execution of the DestroySystem( ) method shall effect the destruction of the referenced virtual system
1068 and all related virtual system configurations, including snapshots.

1069 Table 4 contains requirements for parameters of this method.

1070 **Table 4 – DestroySystem( ) method: Parameters**

Qualifiers	Name	Type	Description/Values
IN	AffectedSystem	CIM_ComputerSystem REF	See 8.2.2.1.
OUT	Job	CIM_ConcreteJob REF	See 8.1.3.2.

1071 #### 8.2.2.1 AffectedSystem parameter

1072 A client shall set a value of the AffectedSystem parameter to refer to the instance of the
1073 CIM_ComputerSystem class that represents the virtual system to be destroyed.

1074 An implementation shall interpret the value of the AffectedSystem parameter to identify the virtual system
1075 that is to be destroyed.

1076 #### 8.2.2.2 Return codes

1077 An implementation shall indicate the result of the method execution by using the return code values speci-
1078 fied in Table 5.

1079 **Table 5 – 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 8.1.3 apply.

1080 ### 8.2.3 CIM_VirtualSystemManagementService.AddResourceSettings( ) method
1081 ### (Conditional)

1082 The implementation of the AddResourceSettings( ) method is conditional.

1083 Condition: The addition and the removal of virtual resources to virtual systems is implemented; see
1084 7.4.6.2.

1085 If the AddResourceSettings( ) method is implemented, the provisions in this subclause apply; in addition
1086 behavior applicable to all extrinsic methods is specified in 8.1.2.

1087 The execution of the AddResourceSettings( ) method shall effect the entry of resource allocation requests
1088 or resource allocations provided through the ResourceSettings[ ] array parameter in the affected virtual
1089 system configuration.

1090 Table 6 contains requirements for parameters of this method.

1091 **Table 6 – AddResourceSettings( ) method: Parameters**

Qualifiers	Name	Type	Description/Values
IN	AffectedConfiguration	CIM_VirtualSystemSettingData REF	See 8.2.3.1.
IN	ResourceSettings[ ]	string	See 8.2.3.2.
OUT	ResultingResourceSettings[ ]	CIM_ResourceAllocationSettingData REF	See 8.2.3.3.
OUT	Job	CIM_ConcreteJob REF	See 8.1.3.2.

1092 #### 8.2.3.1 AffectedConfiguration parameter

1093 A client shall set a value of AffectedConfiguration parameter to refer to the instance of the
1094 CIM_VirtualSystemSettingData class that represents the virtual system configuration that receives new
1095 resource allocations.

1096 An implementation shall interpret the value of the AffectedConfiguration parameter to identify the virtual
1097 system configuration that receives new resource allocations.

1098 #### 8.2.3.2 ResourceSettings[ ] array parameter

1099 A client shall set the ResourceSettings[ ] parameter containing one or more input instances of the
1100 CIM_ResourceAllocationSettingData class as specified in a profile based on DSP1041 and on DSP1043,
1101 such as for example DSP1044 or DSP1047.

1102 If the value of the InstanceID property in any of the input CIM_ResourceAllocationSettingData instances
1103 is other than NULL, that value shall be ignored; however, the remaining values of the input instance shall
1104 be respected as defined in the resource type specific resource allocation profile.

1105 An implementation shall apply the specifications given in 8.1.1.

1106 #### 8.2.3.3 ResultingResourceSettings[ ] array parameter

1107 The implementation shall set the value of the ResultingResourceSettings[ ] array parameter as follows:

1108 • to an array of references to instances of the CIM_ResourceAllocationSettingData class that
1109 represent resource allocations that were obtained during the execution of the method

1110 • to NULL, if the method is executed synchronously and fails, or if the method is executed
1111 asynchronously

1112 #### 8.2.3.4 Return codes

1113 An implementation shall indicate the result of the method execution by using the return code values speci-
1114 fied in Table 7.

1115 **Table 7 – 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.

Value	Description
4096	Method execution is performed asynchronously. The specifications given in 8.1.3 apply.

8.2.4 CIM_VirtualSystemManagementService.ModifyResourceSettings( ) method

The implementation of the ModifyResourceSettings( ) method is conditional.

Condition: The modification of virtual systems and resources is implemented; see 7.4.6.3.

If the ModifyResourceSettings( ) method is implemented, the provisions in this subclause apply; in addition behavior applicable to all extrinsic methods is specified in 8.1.2.

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 8 contains requirements for parameters of this method.

Table 8 – ModifyResourceSettings( ) method: Parameters

Qualifiers	Name	Type	Description/Values
IN	ResourceSettings[ ]	string	See 8.2.4.1.
OUT	ResultingResourceSettings[ ]	CIM_ResourceAllocationSettingData REF	See 8.2.4.2.
OUT	Job	CIM_ConcreteJob REF	See 8.1.3.2.

8.2.4.1 ResourceSettings[ ] parameter

The specifications in 8.1.1 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 8.1.1. 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.

8.2.4.2 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.

8.2.4.3 Return codes

An implementation shall indicate the result of the method execution by using the return code values specified in Table 9.

Table 9 – 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 8.1.3 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.	

8.2.5 CIM_VirtualSystemManagementService.ModifySystemSettings( ) method

The implementation of the ModifySystemSettings( ) method is conditional.

Condition: The modification of virtual systems and resources is implemented; see 7.4.6.3.

If the ModifySystemSettings( ) method is implemented, the provisions in this subclause apply; in addition behavior applicable to all extrinsic methods is specified in 8.1.2.

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

1174 is provided through the SystemSettings parameter override respective values in the instance identified
1175 through the value of the InstanceID property.

1176 Table 10 contains requirements for parameters of this method.

1177 **Table 10 – ModifySystemSettings( ) Method: Parameters**

Qualifiers	Name	Type	Description/Values
IN	SystemSettings	string	See 8.2.5.1.
OUT	Job	CIM_ConcreteJob REF	See 8.1.3.2.

1178 #### 8.2.5.1 SystemSettings parameter

1179 A client shall set the SystemSettings parameter. Any instance of the CIM_VirtualSystemSettingData class
1180 that is passed in as a value of the SystemSettings parameter shall have a valid non-NULL value in the
1181 InstanceID property that identifies a respective instance of the CIM_VirtualSystemSettingData class
1182 existing within the implementation. A client shall obtain such an instance before invoking the
1183 ModifySystemSettings( ) method (for example, by using an extrinsic method or intrinsic CIM operation
1184 that yields a respective instance as a result). For example, the client may use the intrinsic GetInstance( )
1185 CIM operation. The client shall then modify the instance locally so that it reflects the desired modifications
1186 and finally pass it back in as a value of the SystemSettings parameter.

1187 The implementation shall ignore any value of the SystemSettings parameter that does not identify,
1188 through the value of the InstanceID key property, an existing instance of the
1189 CIM_VirtualSystemSettingData class within the implementation.

1190 #### 8.2.5.2 Return codes

1191 An implementation shall indicate the result of the method execution by using the return code values speci-
1192 fied in Table 11.

1193 **Table 11 – 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 8.1.3 apply.

1194 ### 8.2.6 CIM_VirtualSystemManagementService.RemoveResourceSettings( ) method

1195 The implementation of the RemoveResourceSettings( ) method is conditional.

1196 Condition: The addition and the removal of virtual resources to virtual systems is implemented; see
1197 7.4.6.2.

If the RemoveResourceSettings( ) method is implemented, the provisions in this subclause apply; in addition behavior applicable to all extrinsic methods is specified in 8.1.2.

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 12 contains requirements for parameters of this method.

Table 12 – RemoveResourceSettings( ) Method: Parameters

Qualifiers	Name	Type	Description/Values
IN	ResourceSettings[ ]	CIM_ResourceAllocationSettingData REF	See 8.2.6.1.
OUT	Job	CIM_ConcreteJob REF	See 8.1.3.2.

8.2.6.1 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.

8.2.6.2 Return codes

An implementation shall indicate the result of the method execution by using the return code values specified in Table 13.

Table 13 – 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 8.1.3 apply.

8.3 Methods of the CIM_VirtualSystemSnapshotService class

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

8.3.1 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 7.7.1.1.

If the CreateSnapshot( ) method is implemented, the provisions in this subclause apply; in addition behavior applicable to all extrinsic methods is specified in 8.1.2.

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 8.3.1.3).

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 14 contains requirements for parameters of this method.

Table 14 – CreateSnapshot( ) method: Parameters

Qualifiers	Name	Type	Description/Values
IN	AffectedSystem	CIM_ComputerSystem REF	See 8.3.1.1.
IN	SnapshotSettings	string	See 8.3.1.2.
IN	SnapshotType	uint16	See 8.3.1.3.
OUT	ResultingSnapshot	CIM_VirtualSystemSettingData REF	See 8.3.1.4.
OUT	Job	CIM_ConcreteJob REF	See 8.1.3.2.

8.3.1.1 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.

8.3.1.2 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.

8.3.1.3 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).

8.3.1.4 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 8.1.2 to locate the instance of the CIM_VirtualSystemSettingData class that represents the newly created virtual system snapshot.

8.3.1.5 Return codes

An implementation shall indicate the result of the method execution by using the return code values specified in Table 15.

Table 15 – 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 8.1.3 apply.

8.3.2 VirtualSystemSnapshotService.DestroySnapshot( ) method

The implementation of the DestroySnapshot( ) method is conditional.

Condition: Virtual system snapshots are implemented; see 7.7.1.1.

If the DestroySnapshot( ) method is implemented, the provisions in this subclause apply; in addition behavior applicable to all extrinsic methods is specified in 8.1.2.

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 16 contains requirements for parameters of this method.

Table 16 – DestroySnapshot( ) method: Parameters

Qualifiers	Name	Type	Description/Values
IN	AffectedSnapshot	CIM_VirtualSystemSettingData REF	See 8.3.2.1.
OUT	Job	CIM_ConcreteJob REF	See 8.1.3.2.

8.3.2.1 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.

8.3.2.2 Return codes

An implementation shall indicate the result of the method execution using the return code values specified by Table 17.

Table 17 – 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 8.1.3 apply.

8.3.3 VirtualSystemSnapshotService.ApplySnapshot( ) method

The implementation of the ApplySnapshot( ) method is conditional.

Condition: Virtual system snapshots are implemented; see 7.7.1.1.

If the ApplySnapshot( ) method is implemented, the provisions in this subclause apply; in addition behavior applicable to all extrinsic methods is specified in 8.1.2.

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 18 contains requirements for parameters of this method.

Table 18 – ApplySnapshot( ) method: Parameters

Qualifiers	Name	Type	Description/Values
IN	Snapshot	CIM_VirtualSystemSettingData REF	See 8.3.3.1.
OUT	Job	CIM_ConcreteJob REF	See 8.1.3.2.

8.3.3.1 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.

8.3.3.2 Return codes

An implementation shall indicate the result of the method execution by using the return code values specified in Table 19.

Table 19 – 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 8.1.3 apply.

8.4 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( )

8.4.1 CIM_AffectedJobElement

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.2 CIM_ComputerSystem

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.3 CIM_ConcreteJob

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.4 CIM_Dependency

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.5 CIM_ElementCapabilities

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.6 CIM_ElementConformsToProfile

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.7 CIM_HostedDependency

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.8 CIM_HostedService

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.9 CIM_LastAppliedSnapshot

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.10 CIM_MostCurrentSnapshotInBranch

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.11 CIM_ReferencedProfile

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.12 CIM_RegisteredProfile

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.13 CIM_ServiceAffectsElement

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.14 CIM_SnapshotOfVirtualSystem

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.15 CIM_System

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.16 CIM_VirtualSystemManagementCapabilities

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.17 CIM_VirtualSystemManagementService

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.18 CIM_VirtualSystemSnapshotService

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.19 CIM_VirtualSystemSnapshotCapabilities

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

8.4.20 CIM_VirtualSystemSnapshotServiceCapabilities

All operations in the default list in 8.4 shall be implemented as defined in DSP0200.

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

9 Use Cases

This clause contains informative text only.

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

9.1 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 9.2.1, the following conditions are implicitly assumed:

- The client knows the URL of a WBEM service that exposes an implementation of this profile.
- 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.

9.2 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 3 outlines a sample situation that is referenced by some of the use-case descriptions in subsequent subclauses.

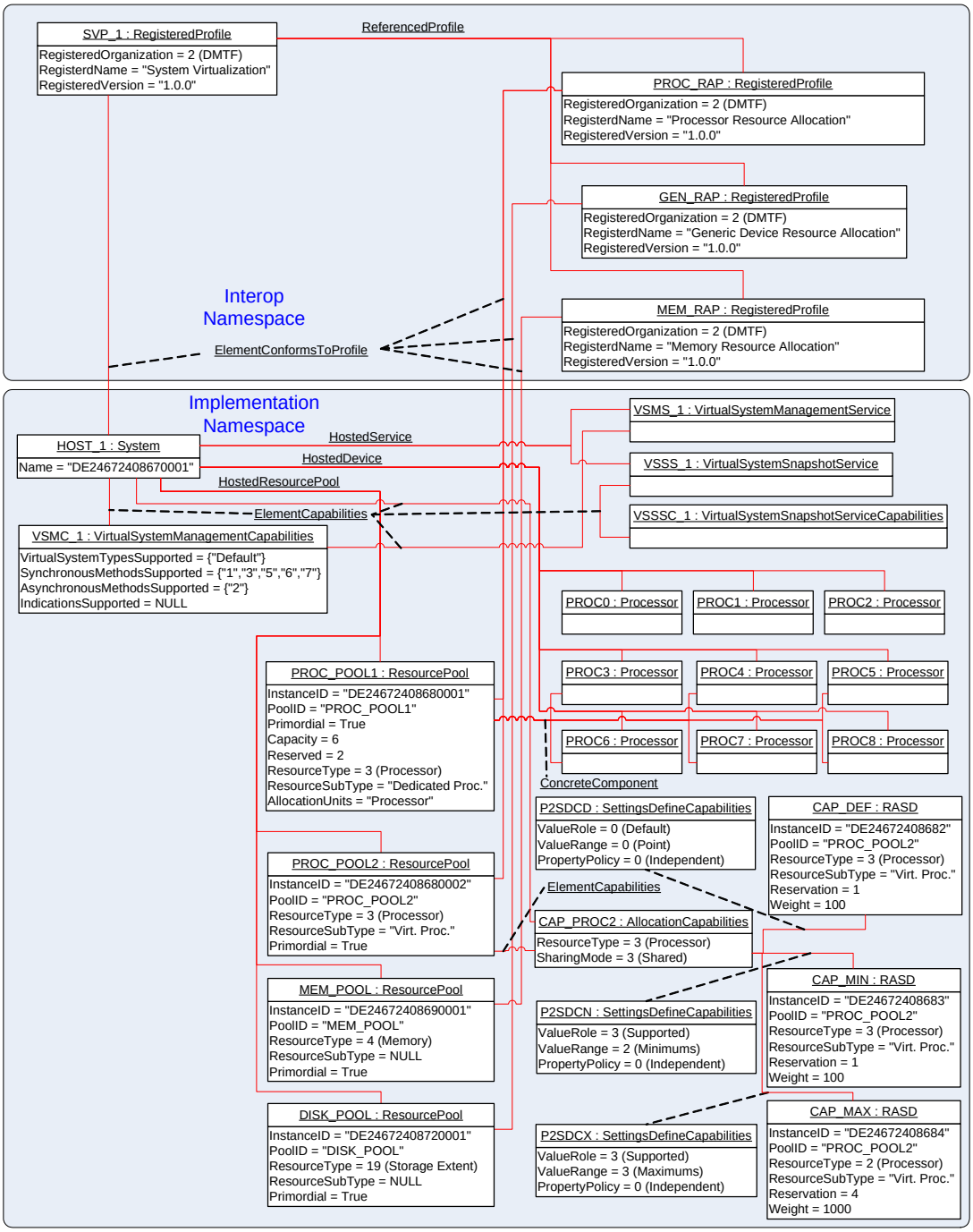

Figure 3 – System Virtualization Profile instance diagram: Discovery, localization, and inspection

9.2.1 SLP-Based discovery of CIM object managers hosting implementations of this 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: This profile 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 this profile is implemented.

9.2.2 Locate conformant implementations using the EnumerateInstances( ) operation

Assumption: The client knows the URL of a WBEM service hosting implementations of this profile (see 9.2.1).

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.

1) 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 this profile.

In the example shown in Figure 3, one instance of the CIM_RegisteredProfile class represents an implementation of this profile; it is tagged SVP_1.

9.2.3 Locate conformant implementations using the ExecuteQuery( ) operation

Assumption: The client knows the URL of a WBEM service hosting implementations of this profile (see 9.2.1).

- 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 this profile.

In the example shown in Figure 3, one instance of the CIM_RegisteredProfile class represents an implementation of this profile; it is tagged SVP_1.

9.2.4 Locate host systems represented by central instances of this profile

Assumption: The client knows a reference to an instance of the CIM_RegisteredProfile class that represents an implementation of this profile (see 9.2.2 or 9.2.3).

- 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 this profile.

In the example shown in Figure 3, one instance of the CIM_RegisteredProfile class represents a host system that is a central and scoping instance of this profile; it is tagged HOST_1.

9.2.5 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 this profile (see 9.2.2 or 9.2.3).

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 this profile.
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 1.

   If the value does not match any name of a resource allocation DMTF management profile scoped by this profile, 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 this profile.

In the example shown in Figure 3, 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 this profile. These instances represent implementations of scoped resource allocation DMTF management profiles:

- The instance tagged PROC_RAP represents an implementation of DSP1044.
- The instance tagged GEN_RAP represents an implementation of DSP1059.
- The instance tagged MEM_RAP represents an implementation of DSP1045.

9.2.6 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 this profile (see 9.2.4).

- 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 3, one instance of the CIM_VirtualSystemManagementService class serves the host system; it is tagged VSMS_1.

9.2.7 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 this profile (see 9.2.4).

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.

3) 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 3, 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.

9.2.8 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 this profile (see 9.2.4).

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.

4) 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.

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

Resource allocation DMTF management profiles are based on DSP1041 that 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 9.2.5).

- 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 9.2.8 to obtain a list of instances of the CIM_ResourcePool class that each represent a resource pool of that particular resource type.

1571 The resulting list is the result for this use case.

1572 **Result:** The client knows a list of instances of the CIM_ResourcePool class, each representing a central
1573 instance of a scoped resource allocation DMTF management profile.

1574 ### 9.2.10 Determine implemented resource types

1575 **Assumption:** The client knows a reference to an instance of the CIM_RegisteredProfile class that
1576 represents an implementation of this profile (see 9.2.2 or 9.2.3).

1577 1) The client locates implementations of DMTF management profiles that are scoped by this profile
1578 (see 9.2.5).

1579 The result is a list of references to instances of the CIM_RegisteredProfile class that represent
1580 implementations of DMTF management profiles that are scoped by this profile.

1581 5) For each instance of CIM_RegisteredProfile, the client obtains the set of instances of the
1582 CIM_ResourcePool class that are central instances of the respective scoped resource allocation
1583 DMTF management profiles and represent a conformant resource pool (see 9.2.9).

1584 The result is a list of instances of the CIM_ResourcePool class that are central instances of
1585 scoped resource allocation DMTF management profiles.

1586 6) The client creates an initially empty list of integer values. For each instance that is a result from
1587 step 5), the client determines whether the value of property ResourceType is already repre-
1588 sented in the list:

1589 – If that value is already contained in the list, the client ignores the element.

1590 – If that value is not yet contained in the list, the client adds a new element to the list with
1591 that value.

1592 **Result:** The client knows a list of integer values, each designating a resource type that is supported by
1593 the implementation.

1594 In the example shown in Figure 3, three instances of the CIM_RegisteredProfile class are associated with
1595 the instance of the CIM_RegisteredProfile class that represents the implementation of this profile. These
1596 instances are central instances of scoped resource allocation DMTF management profiles:

1597 • The instance tagged PROC_RAP represents an implementation of DSP1044.

1598 • The instance tagged GEN_RAP represents an implementation of DSP1059.

1599 • The instance tagged MEM_RAP represents an implementation of DSP1045.

1600 These instances are all associated with respective instances of the CIM_ResourcePool class, indicating
1601 that in this example in all cases the central class profile advertisement methodology is in use:

1602 • The instance tagged PROC_RAP is associated with two instances that represent resource
1603 pools for the allocation of processors. They show a value of 3 (Processor) for the ResourceType
1604 property and are tagged PROC_POOL1 and PROC_POOL2.

1605 • The instance tagged GEN_RAP is associated with one instance that represents a resource pool
1606 for the allocation of virtual disks. It shows a value of 19 (Storage Extent) for the ResourceType
1607 property and is tagged DISK_POOL.

1608 • The instance tagged MEM_RAP is associated with one instance that represents a resource pool
1609 for the allocation of memory. It shows a value of 4 (Memory) for the ResourceType property and
1610 is tagged MEM_POOL.

1611 The resulting list of integer values is {"3","4","19"} and designates the implemented resource types
1612 3 (Processor), 4 (Memory), and 19 (Storage Extent).

9.2.11 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 this profile (see 9.2.4).

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.

7) 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 7) 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.

8) For each instance of the list resulting from step 7), 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 7).

9) From the list obtained in step 8), 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.

10) 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 9).
    - 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 3, 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 DSP1043 shows only one associated element for each resource type.
- With step 7), the client reduces the result set to the one element that describes allocation capabilities processors. This instance is tagged CAP_PROC1.
- With steps 8) and 9), 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 10), 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.

9.2.12 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.

9.2.13 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.

11) 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.

12) 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.

13) For each instance that is a result from step 12), 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 DSP1043 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.

14) For each association instance obtained in step 13), 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 13).

    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.

9.2.14 Determine implementation class specifics

This profile 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.

15) 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.

9.2.15 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 9.2.13 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".

16) The client applies use case 9.2.14 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.

9.2.16 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 this profile and represents a host system (see 9.2.4).

- 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.

9.3 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 9.2.6).

9.3.1 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.

9.3.1.1 Define virtual system based on input and reference virtual system configuration

Assumption: No assumption is made beyond the general assumption specified in 9.3.

1) The client invokes the DefineSystem( ) method (see 8.2.1) 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.
1) 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 4 shows the representation of a virtual system that was defined using an "Input" virtual system and a "Reference" virtual system configuration.

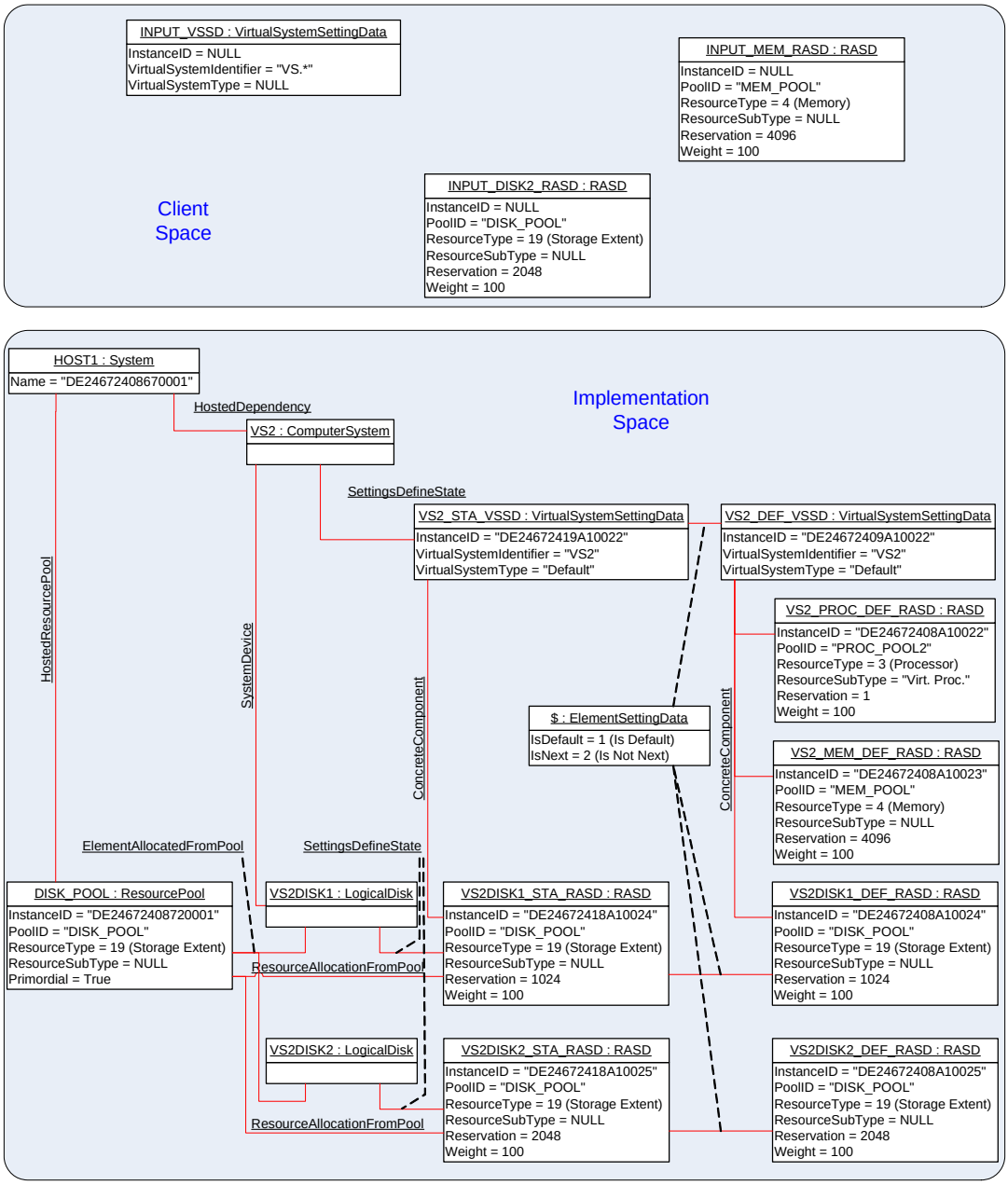

Figure 4 – Virtual system configuration based on input virtual system configurations and implementation defaults

The new virtual system is represented by an instance of the CIM_ComputerSystem class that is tagged VS2. The right side of Figure 4 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 4. 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.

9.3.1.2 Define virtual system with implementation-specific properties

Assumption: No assumption is made beyond the general assumption specified in 9.3.

- The client performs use case 9.3.1.1 using an input configuration only. While preparing the input virtual system configuration, the client applies use case 9.2.14 to determine the implementation class of the CIM_VirtualSystemSettingData class and use case 9.2.15 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.

9.3.2 Virtual system modification

This clauses describes a set of usecases that modify virtual systems or virtual system configurations.

9.3.2.1 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 (respective use cases are described in DSP1057).

17) The client makes conformant changes to the instance of the CIM_VirtualSystemSettingData class. In particular, the client must not modify key properties.

18) The client invokes the ModifySystemSettings( ) method (see 8.2.5) on the virtual system management service. The value of the SystemSettings parameter is the modified instance from step 17).

19) 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.

9.3.2.2 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.

20) The client invokes the AddResourceSettings( ) method (see 8.2.3) 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).

2) 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.

9.3.2.3 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 23).

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 21) and 22), and proceeds with step 23). If the client intends to modify the virtual resource definition, the client continues with step 21).

21) 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.

22) The client selects from the result set of step 21) 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.

23) 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 22).

The result is the instance of the CIM_ResourceAllocationSettingData class that represents the virtual resource definition.

24) 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 24) repetitively, preparing a set of resource allocation change requests that subsequently are applied as one atomic operation.

25) The client invokes the ModifyResourceSettings( ) method (see 8.2.4) 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 24).

26) 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 5 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 5 are involved in the example of a processor resource modification that is described following the figure.

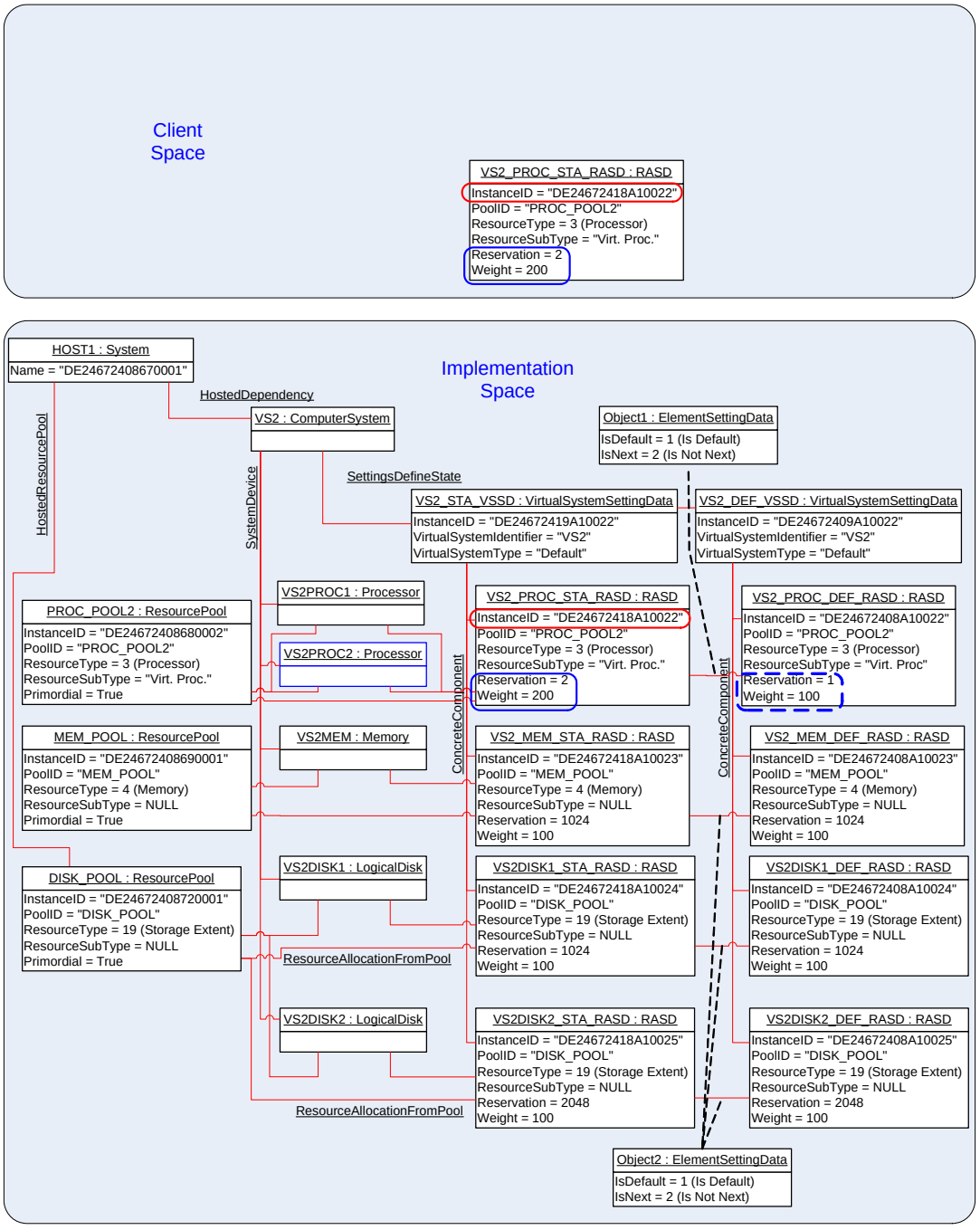

Figure 5 – Virtual system resource modification

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.

9.3.2.4 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. See DSP1057, clause 9, for respective use cases.

1) The client invokes the RemoveResourceSettings( ) method (see 8.2.6) 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.

9.3.3 Destroy virtual system

Assumption: The client knows a reference to an instance of the CIM_ComputerSystem class that represents a virtual system (see 9.2.16).

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.

9.4 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 6 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 DSP1057.

The sequence of events that yield the situation shown in Figure 6 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.

27) At a time after T0 but before T1, the virtual system is activated.

28) 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.

29) 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.

30) 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.

31) 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.

32) 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.

33) 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.

34) 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.

35) At time T3, the situation is as shown in Figure 6.

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 9.2.6).

Current Time: T3 > T2 > T1 > T0

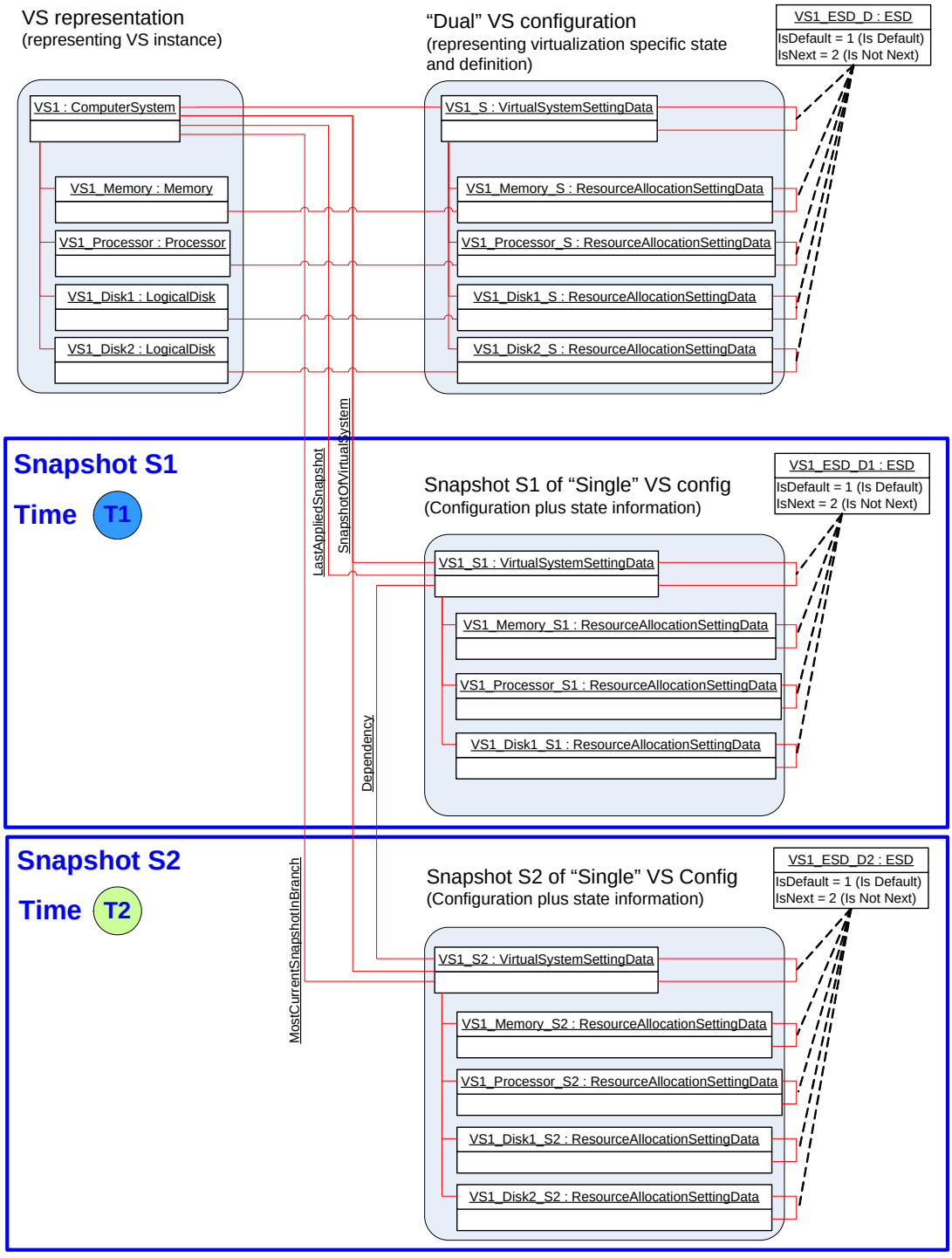

Figure 6 – System Virtualization Profile: Snapshot example

9.4.1 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 this profile; (see 9.2.4).

- 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 3, one instance of the CIM_VirtualSystemSnapshotService class serves the host system; it is tagged VSSS_1.

9.4.2 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 9.4.1).

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.

36) 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 3, one instance of the CIM_VirtualSystemSnapshotServiceCapabilities class is associated with the host system; it is tagged VSSSC_1.

9.4.3 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 9.2.16). 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).

37) 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 6, 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 6 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).

9.4.4 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 9.2.16).

- 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 6, the instances tagged VS_S1 and VS1_S2 of the CIM_VirtualSystemSettingData class represent snapshots S1 and S2.

9.4.5 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 6, 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.

9.4.6 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 9.2.16).

- 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 6, 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.

9.4.7 Locate dependent snapshots

Assumption: The client knows a reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot (see 9.4.4).

- 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 6, the instance tagged VS_S2 represents snapshot S2, which is dependent on snapshot S1, which is represented by the instance tagged VS_S1.

9.4.8 Locate parent snapshot

Assumption: The client knows a reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot (see 9.4.4).

- 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 6, the instance tagged VS_S1 represents snapshot S1, which is the parent of snapshot S2, which is represented by the instance tagged VS_S2.

9.4.9 Apply snapshot

Assumption: The client knows a reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot (see 9.4.3 or 9.4.4). 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 9.4.5). 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 6, the situation is depicted at time T3, immediately after the activation of snapshot S2 within virtual system VS1.

9.4.10 Destroy snapshot

Assumption: The client knows the reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot (see 9.2.16).

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.

10 CIM elements

Table 20 lists CIM elements that are defined or specialized for this profile. Each CIM element shall be implemented as described in Table 20. The CIM Schema descriptions for any referenced element and its sub-elements apply.

Clauses 7 ("Implementation") and 8 ("Methods") may impose additional requirements on these elements.

Table 20 – CIM Elements: System Virtualization Profile

Element Name	Requirement	Description
CIM_AffectedJobElement	Conditional	See 10.1.
CIM_ConcreteJob	Conditional	See 10.2.
CIM_Dependency	Conditional	See 10.3.
CIM_ElementCapabilities (Host system)	Mandatory	See 10.4.
CIM_ElementCapabilities (Virtual system management service)	Mandatory	See 10.5.
CIM_ElementCapabilities (Virtual system snapshot service)	Conditional	See 10.6.
CIM_ElementCapabilities (Snapshots of virtual systems)	Conditional	See 10.7.
CIM_ElementConformsToProfile	Mandatory	See 10.8.
CIM_HostedDependency	Mandatory	See 10.9.
CIM_HostedService (Virtual system management service)	Conditional	See 10.10.
CIM_HostedService (Virtual system snapshot service)	Conditional	See 10.11.
CIM_LastAppliedSnapshot	Conditional	See 10.12.
CIM_MostCurrentSnapshotInBranch	Conditional	See 10.13.
CIM_ReferencedProfile	Conditional	See 10.14.
CIM_RegisteredProfile	Mandatory	See 10.15.
CIM_ServiceAffectsElement (Virtual system management service)	Conditional	See 10.16.
CIM_ServiceAffectsElement (Virtual system snapshot service)	Conditional	See 10.17.
CIM_SnapshotOfVirtualSystem	Conditional	See 10.18.
CIM_System	Mandatory	See 10.19.
CIM_VirtualSystemManagementCapabilities	Mandatory	See 10.20.
CIM_VirtualSystemManagementService	Conditional	See 10.21.
CIM_VirtualSystemSettingData (Input)	Conditional	See 10.22.

Element Name	Requirement	Description
CIM_VirtualSystemSettingData (Snapshot)	Conditional	See 10.23.
CIM_VirtualSystemSnapshotCapabilities	Conditional	See 10.24.
CIM_VirtualSystemSnapshotService	Optional	See 10.25.
CIM_VirtualSystemSnapshotServiceCapabilities	Conditional	See 10.26.

10.1 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 21 contains the requirements for elements of this association.

Table 21 – Association: CIM_AffectedJobElement

Elements	Requirement	Notes
AffectedElement	Mandatory	**Key:** See 8.1.2. **Cardinality:** *
AffectingElement	Mandatory	**Key:** See 8.1.2. **Cardinality:** 1
ElementEffects[ ]	Mandatory	See 8.1.2.

10.2 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 22 contains requirements for elements of this class.

Table 22 – Class: CIM_ConcreteJob

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
JobState	Mandatory	See 8.1.2.
TimeOfLastStateChange	Mandatory	See 8.1.2.

10.3 CIM_Dependency

The implementation of the CIM_Dependency association is conditional.

Condition: Virtual system snapshots are implemented; see 7.7.1.1.

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 23 contains requirements for elements of this class.

Table 23 – 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

10.4 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 24 contains requirements for elements of this association.

Table 24 – 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

10.5 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 7.4.6.1)

- Virtual resource addition and removal (see 7.4.6.2)
- Virtual system and resource modification (see 7.4.6.3)

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 25 contains requirements for elements of this association.

Table 25 – 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

10.6 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 7.7.1.1.

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 26 contains requirements for elements of this association.

Table 26 – 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

10.7 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 7.7.1.1.

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 27 contains requirements for elements of this association.

Table 27 – 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

10.8 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 this profile with instances of the CIM_System class that represent a host system that is a central and scoping instance of this profile.

Table 28 contains requirements for elements of this association.

Table 28 – Association: CIM_ElementConformsToProfile

Elements	Requirement	Notes
ConformantStandard	Mandatory	**Key:** Reference to an instance of the CIM_RegisteredProfile class that represents an implementation of this profile **Cardinality:** 1
ManagedElement	Mandatory	**Key:** Reference to an instance of the CIM_ System class that represents a host system **Cardinality:** *

10.9 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 29 contains requirements for elements of this association.

Table 29 – 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:** *

10.10 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 7.4.6.1)
- Vvirtual resource addition and removal (see 7.4.6.2)
- Virtual system and resource modification (see 7.4.6.3)

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_VirtualSystemManagementService class that represents the virtual system management service that is hosted by a host system.

Table 30 contains requirements for elements of this association.

Table 30 – 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

10.11 CIM_HostedService (Virtual system snapshot service)

The implementation of the CIM_HostedService association is conditional.

Condition: Virtual system snapshots are implemented; see 7.7.1.1.

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 31 contains requirements for elements of this association.

Table 31 – 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

10.12 CIM_LastAppliedSnapshot

The implementation of the CIM_LastAppliedSnapshot association is conditional.

Condition: Virtual system snapshots are implemented; see 7.7.1.1.

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 32 contains requirements for elements of this association.

Table 32 – 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

10.13 CIM_MostCurrentSnapshotInBranch

The implementation of the CIM_MostCurrentSnapshotInBranch association is conditional.

Condition: Virtual system snapshots are implemented; see 7.7.1.1.

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 33 contains requirements for elements of this association.

Table 33 – 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

10.14 CIM_ReferencedProfile

The implementation of the CIM_ReferencedProfile association is conditional.

Condition: Resource virtualization profiles such as DSP1059 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 this profile 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 34 contains requirements for elements of this association.

Table 34 – Association: CIM_ReferencedProfile

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Reference to an instance of the CIM_RegisteredProfile that represents an implementation of this profile **Cardinality:** 1

Dependent	Mandatory	**Key:** Reference to an instance of the CIM_RegisteredProfile class that represents an implementation of a resource allocation profile **Cardinality:** *

10.15 CIM_RegisteredProfile

An implementation shall use an instance of the CIM_RegisteredProfile class to represent an implementation of this profile.

Table 35 contains requirements for elements of this class.

Table 35 – 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 this profile ("1.0.0").

10.16 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 7.4.6.1)
- Virtual resource addition and removal (see 7.4.6.2)
- Virtual system and resource modification (see 7.4.6.3)

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 36 contains requirements for elements of this association.

Table 36 – 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

10.17 CIM_ServiceAffectsElement (Virtual system snapshot service)

The implementation of the CIM_ServiceAffectsElement association is conditional.

Condition: Virtual system snapshots are implemented; see 7.7.1.1.

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 37 contains requirements for elements of this association.

Table 37 – 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

10.18 CIM_SnapshotOfVirtualSystem

The implementation of the CIM_SnapshotOfVirtualSystem association is conditional.

Condition: Virtual system snapshots are implemented; see 7.7.1.1.

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 38 contains requirements for elements of this association.

Table 38 – 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:** *

10.19 CIM_System

An implementation shall use an instance of a concrete subclass of the CIM_System class to represent a host system.

Table 40 contains requirements for elements of this class.

Table 39 – Class: CIM_VirtualSystemManagementCapabilities

Elements	Requirement	Notes
CreationClassName	Mandatory	**Key**.
Name	Mandatory	**Key**

10.20 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 40 contains requirements for elements of this class.

Table 40 – Class: CIM_VirtualSystemManagementCapabilities

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
VirtualSystemTypesSupported[ ]	Optional	See 7.4.2.
SynchronousMethodsSupported[ ]	Optional	See 7.4.3.
AsynchronousMethodsSupported[ ]	Optional	See 7.4.4.
IndicationsSupported[ ]	Optional	See.7.4.5.

10.21 CIM_VirtualSystemManagementService

The implementation of the CIM_VirtualSystemManagementService class is conditional.

Condition: Any of the following is implemented:

- Virtual system definition and destruction (see 7.4.6.1)
- Virtual resource addition and removal (see 7.4.6.2)
- Virtual system and resource modification (see 7.4.6.3)

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 41 contains requirements for elements of this class.

2396 **Table 41 – Class: CIM_VirtualSystemManagementService**

Elements	Requirement	Notes
CreationClassName	Mandatory	**Key**
Name	Mandatory	**Key**
SystemCreationClassName	Mandatory	**Key**
SystemName	Mandatory	**Key**
AddResourceSettings( )	Conditional	See 8.2.3.
DefineSystem( )	Conditional	See 8.2.1.
DestroySystem( )	Conditional	See 8.2.2.
ModifyResourceSettings( )	Conditional	See 8.2.4.
ModifySystemSettings( )	Conditional	See 8.2.5.
RemoveResourceSettings( )	Conditional	See 8.2.6.

2397 ## 10.22 CIM_VirtualSystemSettingData (Input)

2398 The implementation of the CIM_VirtualSystemSettingData class for input is conditional.

2399 Condition: Any of the following is implemented:

2400 • Virtual system definition and destruction (see 7.4.6.1)

2401 • Virtual resource addition and removal (see 7.4.6.2)

2402 • Virtual system and resource modification (see 7.4.6.3)

2403 If the CIM_VirtualSystemSettingData class is implemented for input, the provisions in this subclause
2404 apply.

2405 An instance of the CIM_VirtualSystemSettingData class shall be used to represent input data for a virtual
2406 system's definitions and modifications.

2407 Table 42 contains requirements for elements of this class.

2408 **Table 42 – Class: CIM_VirtualSystemSettingData (Input)**

Elements	Requirement	Notes
InstanceID	Mandatory	**Key** (Input): See 7.5.1.
ElementName	Optional	See 7.5.2 .
VirtualSystemIdentity	Optional	See 7.5.3.
VirtualSystemType	Optional	See 7.5.4.

2409 ## 10.23 CIM_VirtualSystemSettingData (Snapshot)

2410 The implementation of the CIM_VirtualSystemSettingData class for the representation of snapshots of vir-
2411 tual systems is conditional.

2412 Condition: Virtual system snapshots are implemented; see 7.7.1.1.

2413 If the CIM_VirtualSystemSettingData class is implemented for the representation of snapshots, the
2414 provisions in this subclause apply.

2415 An instance of the CIM_VirtualSystemSettingData class shall be used to represent snapshots of virtual
2416 systems.

2417 Table 43 contains requirements for elements of this class.

2418 **Table 43 – 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.		

2419 10.24 CIM_VirtualSystemSnapshotCapabilities

2420 The implementation of the CIM_VirtualSystemSnapshotCapabilities class is optional.

2421 If the CIM_VirtualSystemSnapshotCapabilities class is implemented, the provisions in this subclause
2422 apply.

2423 The implementation of the optional CIM_VirtualSystemSnapshotCapabilities class is specified only if vir-
2424 tual system snapshots are implemented; (see 7.7.1.1).

2425 An instance of the CIM_VirtualSystemSnapshotCapabilities class may be used to represent the current
2426 applicability of snapshot-related services to one virtual system.

Table 44 contains requirements for elements of this class.

Table 44 – Class: CIM_VirtualSystemSnapshotCapabilities

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
SnapshotTypesEnabled[ ]	Mandatory	See 7.7.5.1.
GuestOSNotificationEnabled[ ]	Optional	See 7.7.5.2.

10.25 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 7.7.1.1).

An instance of the CIM_VirtualSystemSnapshotService class shall be used to represent the virtual system snapshot service available at a host system.

Table 45 contains requirements for elements of this class.

Table 45 – Class: CIM_VirtualSystemSnapshotService

Elements	Requirement	Notes
CreationClassName	Mandatory	**Key**
Name	Mandatory	**Key**
SystemCreationClassName	Mandatory	**Key**
SystemName	Mandatory	**Key**
CreateSnapshot( )	Conditional	See 8.3.1.
DestroySnapshot( )	Conditional	See 8.3.2.
ApplySnapshot( )	Conditional	See 8.3.3.

10.26 CIM_VirtualSystemSnapshotServiceCapabilities

The implementation of the CIM_VirtualSystemSnapshotServiceCapabilities class is conditional.

Condition: Virtual system snapshots are implemented; see 7.7.1.1.

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 46 contains requirements for elements of this class.

Table 46 – Class: CIM_VirtualSystemSnapshotServiceCapabilities

Elements	Requirement	Notes

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
SynchronousMethodsSupported[ ]	Conditional	See 7.7.1.2
AsynchronousMethodsSupported[ ]	Conditional	See 7.7.1.2
SnapshotTypesSupported[ ]	Mandatory	See 7.7.1.2

2447

ANNEX A
(Informative)

Change Log

Version	Date	Description
1.0.0a	2007/08/03	Released as preliminary standard
1.0.0	2009/07/10	Final standard

ANNEX B
(Informative)

Acknowledgements

The authors wish to acknowledge the following people.

Editor:

- Michael Johanssen – IBM

Contributors:

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