Document Number: DSP1042

Date: 2007-08-27

Version: 1.0.0a

System Virtualization Profile

Document Type: Specification

Document Status: Preliminary Standard

Document Language: E

28 # CONTENTS

Figures

Tables

250

Foreword

The *System Virtualization Profile* (DSP1042) 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

The *System Virtualization 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.

2.1 Approved References

DMTF DSP0004, *CIM Infrastructure Specification 2.3.0*

DMTF DSP0200, *CIM Operations over HTTP 1.2.0*

DMTF DSP0201, *Specification for the Representation of CIM in XML 2.2.0*

DMTF DSP1000, *Management Profile Specification Template 1.0*

DMTF DSP1001, *Management Profile Specification Usage Guide 1.0*

DMTF DSP1012, *Boot Control Profile 1.0*

DMTF DSP1022, *CPU Profile 1.0*

DMTF DSP1027, *Power State Management Profile 1.0*

DMTF DSP1033, *Profile Registration Profile 1.0*

DMTF DSP1041, *Resource Allocation Profile 1.0*

DMTF DSP1043, *Allocation Capabilities Profile 1.0*

DMTF DSP1052, *Computer System Profile 1.0*

DMTF DSP1057, *Virtual System Profile 1.0*

2.2 References under Development

DMTF DSP1044, *Processor Device Resource Virtualization Profile 0.7*

DMTF DSP1045, *Memory Resource Virtualization Profile 0.7*

DMTF DSP1047, *Block Based Storage Resource Virtualization Profile 0.2*

DMTF DSP1048, *Memory Resource Virtualization Profile 0.7*

DMTF DSP1049, *Storage Adapter Resource Virtualization Profile 0.7*

DMTF DSP1059, *Generic Device Resource Virtualization Profile 0.9*

2.3 Other References

ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*

OMG UMLINF2.0, *Unified Modeling Language: Infrastructure*

OMG UMLSUP2.0, *Unified Modeling Language: Superstructure*

OPENSLP RFC2608, *RFC Service Location Protocol Version 2*

3 Terms and Definitions

For the purposes of this document, the following terms and definitions and the terms defined in DSP1001 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

virtualization platform

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

4 Symbols and Abbreviated Terms

The following symbols and abbreviations are used in this document.

4.1

CIMOM

CIM object manager

4.2

RASD

resource allocation setting data

4.3

SLP

service location protocol

368 **4.4**
369 **VS**
370 virtual system

371 **4.5**
372 **VSSD**
373 virtual system setting data

374 5 Synopsis

375 **Profile Name:** *System Virtualization*

376 **Version:** 1.0.0a

377 **Organization:** DMTF

378 **CIM Schema Version:** 2.16

379 **Central Class:** CIM_System

380 **Scoping Class:** CIM_System

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

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

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

389 **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.
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.

Profile Name	Organization	Version	Relationship	Description
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 (Informative)

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 DMTF Management Profile Relationships

A client that is exploiting system virtualization facilities specified by the *System Virtualization 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.

The *System Virtualization Profile* complements the *Virtual System Profile*.

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

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

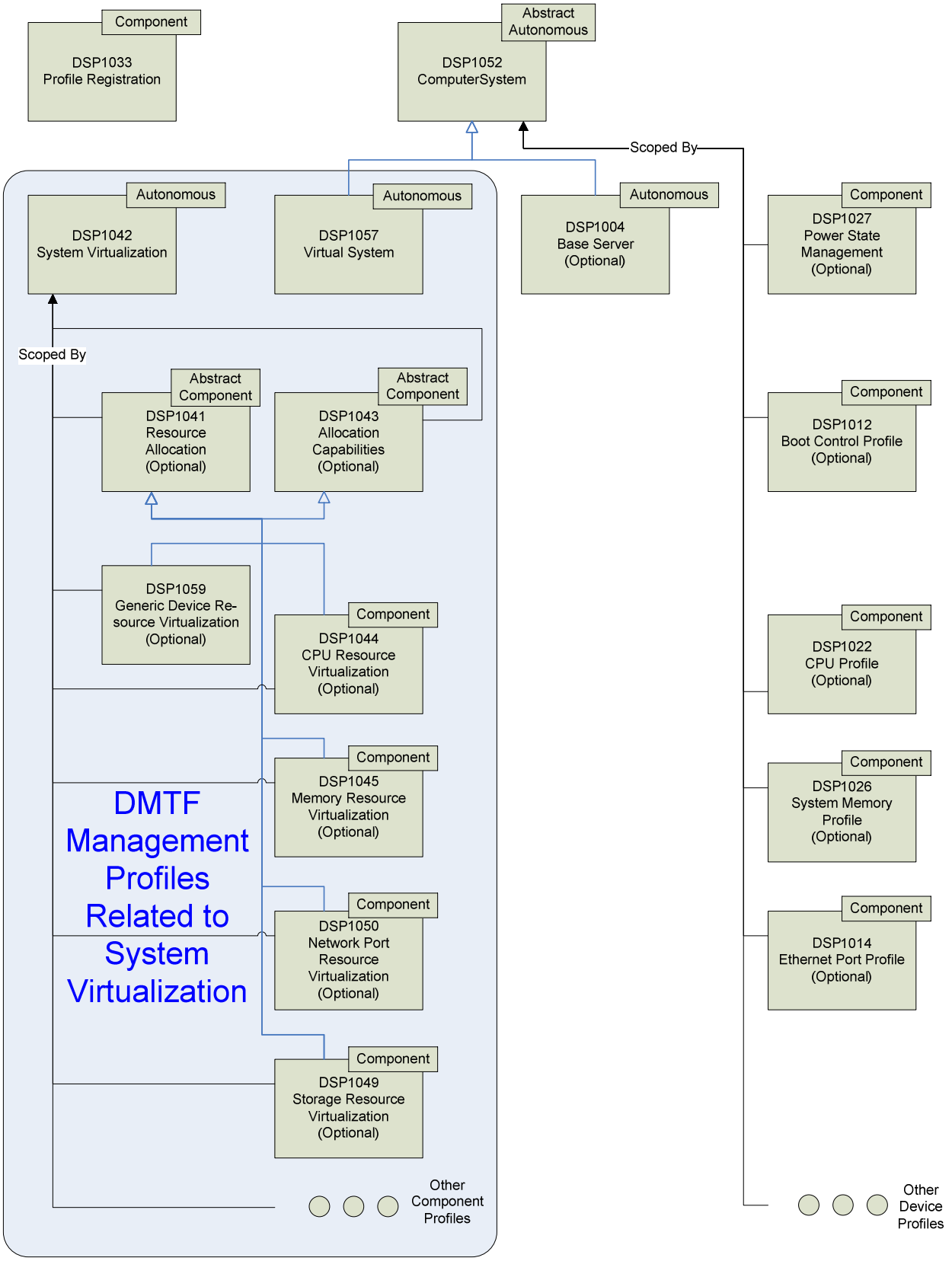


Figure 1 – DMTF Management Profiles Related to System Virtualization

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

- *System Virtualization Profile*

  The *System Virtualization Profile* enables the inspection of host systems, their resources, their capabilities, and their services for creation and manipulation of virtual systems.

- *Virtual System Profile*

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

- Resource-type-specific profiles

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

- Resource allocation profiles

  Resource allocation profiles enable the inspection and management of resource allocation requests, allocated resources, and resources available for allocation. Resource allocation profiles are based on the abstract *Resource Allocation Profile* and on the abstract *Allocation Capabilities Profile*. Resource allocation profiles are scoped by the *System Virtualization 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 the *Processor Device Resource Virtualization Profile* and the *Storage Adapter Resource Virtualization Profile*.

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

6.2 System Virtualization Class Schema

Figure 2 shows the complete class schema of the *System Virtualization 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.

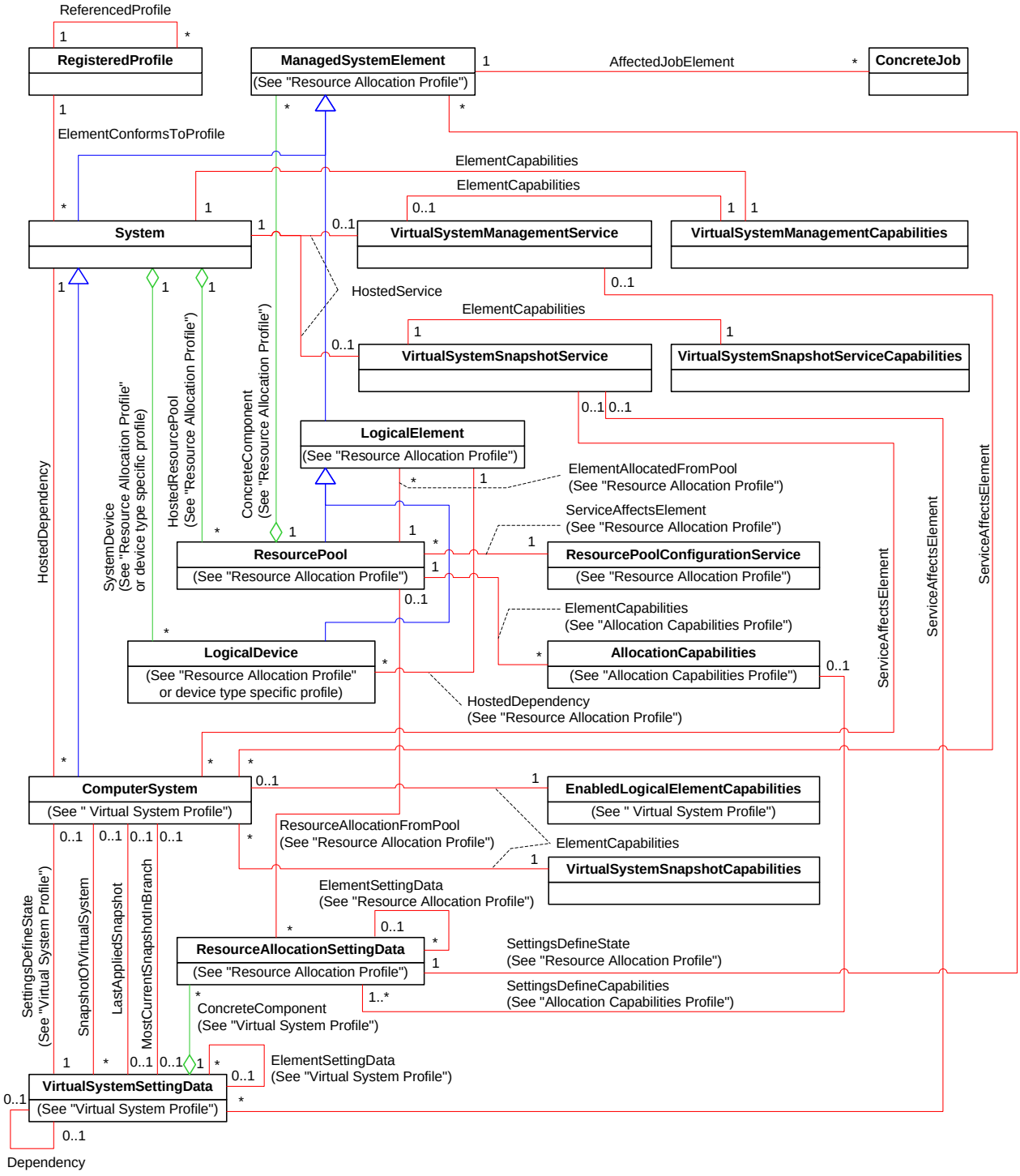


Figure 2 – *System Virtualization Profile*: Class Diagram

The *System Virtualization 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 the *Virtual System Profile*)
- 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

The *System Virtualization Profile* extends the use of virtual system configurations. The *Virtual System Profile* defines a virtual system configuration as one top-level instance of the CIM_VirtualSystemSetting-

Data class that aggregates zero or more instances of the CIM_ResourceAllocationSettingData class through the CIM_VirtualSystemSettingDataComponent association.

The *Virtual System Profile* defines the concept of virtual system configurations and applies it to the following types of virtual system configurations:

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

The *System Virtualization 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.

The *System Virtualization 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

The *Profile Registration Profile* describes how an implementation of a profile shall advertise that a profile is implemented.

7.2.1 System Virtualization Profile

The implementation of the *System Virtualization 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 (Conditional)

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 the *System Virtualization Profile* through an instance of the CIM_ReferencedProfile association.

Resource-allocation DMTF management profiles are based on the *Resource Allocation Profile* and the *Allocation Capabilities Profile*. The resource-allocation DMTF management profiles that are scoped by this profile are listed in Table 1, starting with the *Processor Device Resource Virtualization Profile*.

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

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

The *System Virtualization Profile* strengthens the requirements for the representation of virtual system configurations specified by the *Virtual System Profile* 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 the *Virtual System Profile*. 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 (Optional)

The VirtualSystemTypesSupported[ ] array property should be supported. 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 supported virtual system types, but internally still may exhibit different types of virtual systems.

7.4.3 CIM_VirtualSystemManagementCapabilities.SynchronousMethodsSupported[ ] Array Property (Optional)

The SynchronousMethodsSupported[ ] array property should be supported. Array values shall designate the set of methods of the CIM_VirtualSystemManagementService class that are supported with synchronous behavior only. A NULL value or an empty value set shall be used to indicate that no methods are supported 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 (Optional)

The AsynchronousMethodsSupported[ ] array property should be supported. Array values shall designate the set of methods of the CIM_VirtualSystemManagementService class that are supported with synchronous and potentially with asynchronous behavior. A NULL value or an empty value set shall be used to indicate that no methods are supported 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 (Optional)

The IndicationsSupported[ ] array property should be supported. Array values shall designate the set of types of indications that are supported. A NULL value or an empty value set shall be used to indicate that indications are not supported.

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 supported; nevertheless synchronous and asynchronous behavior may be mixed.

7.4.6.1 Support of Virtual System Definition and Destruction

If virtual system definition and destruction are supported, 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 supported, 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 Support of Virtual Resource Addition and Removal

If the addition and removal of virtual resources to or from virtual systems is supported, 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 supported, 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 Support of Virtual System and Resource Modification

If the modification of virtual systems and virtual resources is supported, 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 supported, 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

The *System Virtualization 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 (Optional)

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

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

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

7.5.3 CIM_VirtualSystemSettingData.VirtualSystemIdentifier Property (Optional)

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

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

Some implementations may accept an implementation-dependent pattern that controls the assignment of a value to the VirtualSystemIdentifier property. For example, an implementation might interpret a regular

764 expression like “^VM\d{1,6}\s” to assign a value to the VirtualSystemIdentifier property that starts with the
765 letters “VM” and is followed by at least one and not more than six digits.

766 7.5.4 CIM_VirtualSystemSettingData.VirtualSystemType Property (Optional)

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

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

777 7.6 Virtual Resource Definition and Modification

778 The *System Virtualization Profile* specifies how to define and modify virtual resources using methods of
779 the virtual system management service. In these method specifications, the
780 CIM_ResourceAllocationSettingData class is used for parameterization of resource allocation specific
781 properties. For specifications that define the use of the CIM_ResourceAllocationSettingData class, see
782 the *Resource Allocation Profile*, the *Allocation Capabilities Profile*, and profiles that specialize these (for
783 example, the *Generic Device Resource Virtualization Profile*). The *Resource Allocation Profile* describes
784 the use of the CIM_ResourceAllocationSettingData class, and the *Allocation Capabilities Profile* intro-
785 duces the concept of allowing a client to determine the acceptable value sets for values of properties of
786 the CIM_ResourceAllocationSettingData class in virtual resource definition and modification requests.

787 7.7 Virtual System Snapshots (Optional)

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

789 7.7.1 Virtual System Snapshot Service and Capabilities

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

792 7.7.1.1 Support of Virtual System Snapshots (Optional)

793 The support of virtual system snapshots is optional. Support includes the creation, destruction, and
794 application of virtual system snapshots.

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

796 • the CIM_VirtualSystemSnapshotService class shall be implemented and the following methods
797 shall be supported:

798 – CreateSnapshot( ), for at least one type of snapshot

799 – DestroySnapshot( )

800 – ApplySnapshot( )

801 • There shall be exactly one instance of the CIM_VirtualSystemSnapshotService class associated
802 to the central instance of this profile through an instance of the CIM_HostedService association.

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

7.7.1.2 CIM_VirtualSystemSnapshotServiceCapabilities Class (Conditional)

Support of the CIM_VirtualSystemSnapshotServiceCapabilities class is conditional with respect to the support of virtual system snapshots (see 7.7.1.1).

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

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

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

The support 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 supported 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 support 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 supported 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 (Conditional)

Virtual snapshot representation is conditional with respect to the support of virtual system snapshots (see 7.7.1.1).

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 (Conditional)

Designation of the last applied snapshot of a virtual system is conditional with respect to the support of virtual system snapshots (see 7.7.1.1).

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 (Conditional)

Designation of the most current snapshot in a branch of snapshots taken of a virtual system is conditional with respect to the support of virtual system snapshots (see 7.7.1.1).

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

848 • A virtual system snapshot is applied to a virtual system.

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

851 • A virtual system snapshot is captured from a virtual system.

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

854 7.7.5 Virtual System Snapshot Capabilities (Optional)

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

857 7.7.5.1 CIM_VirtualSystemSnapshotCapabilities.SnapshotTypesEnabled[ ] Array Property

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

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

863 7.7.5.2 CIM_VirtualSystemSnapshotCapabilities.GuestOSNotificationEnabled Property (Optional)

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

870 8 Methods

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

873 8.1 General Behavior of Extrinsic Methods

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

875 8.1.1 Resource Allocation Requests

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

880 The use of the CIM_ResourceAllocationSettingData class as input for operations is specified in the
881 *Resource Allocation Profile*.

882 One instance of the CIM_ResourceAllocationSettingData class may affect one virtual resource or a coher-
883 ent set of virtual resources. For example, one instance of CIM_ResourceAllocationSettingData that has
884 the value of the ResourceType property set to 3 (Processor) and the value of the VirtualQuantity property
885 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 the implementation does not support a method, it shall set a return value of 1 (Not Supported).

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 supported for asynchronous execution within an implementation are modeled in 7.4.4.
- Elements that convey information about which methods of the CIM_VirtualSystemSnapshotService class are supported for asynchronous execution within an implementation are modeled in 7.7.1.1.
- Elements that convey information about whether a method is executed asynchronously are modeled in 8.1.2.

8.1.3.2 Job Parameter

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

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

8.2 Methods of the VirtualSystemManagementService Class

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

8.2.1 CIM_VirtualSystemManagementService.DefineSystem( ) Method (Conditional)

Behavior applicable to all extrinsic methods is specified in 8.1.2.

The implementation of the DefineSystem( ) method is conditional with respect to support of the definition and destruction of virtual systems (see 7.4.6.1).

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

Table 2 contains requirements for parameters of this method.

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. The *Resource Allocation Profile*, the *Generic Device Resource Virtualization Profile*, 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 the *Virtual System Profile*).

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 (Conditional)

Behavior applicable to all extrinsic methods is specified in 8.1.2.

The implementation of the DestroySystem( ) method is conditional with respect to the support of the definition and destruction of virtual systems (see 7.4.6.1).

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

Table 4 contains requirements for parameters of this method.

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.

8.2.2.1 AffectedSystem Parameter

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

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

8.2.2.2 Return Codes

An implementation shall indicate the result of the method execution by using the return code values specified in Table 5.

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.

8.2.3 CIM_VirtualSystemManagementService.AddResourceSettings( ) Method (Conditional)

Behavior applicable to all extrinsic methods is specified in 8.1.2.

The implementation of the AddResourceSettings( ) method is conditional with respect to the support of the addition and the removal of virtual resources to virtual systems (see 7.4.6.2).

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

Table 6 contains requirements for parameters of this method.

Table 6 – AddResourceSettings( ) Method: Parameters

Qualifiers	Name	Type	Description/Values
IN	AffectedConfiguration	CIM_VirtualSystemSettingData REF	See 8.2.3.1.

Qualifiers	Name	Type	Description/Values
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.

8.2.3.1 AffectedConfiguration Parameter

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

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

8.2.3.2 ResourceSettings[ ] Array Parameter

A client shall set the ResourceSettings[ ] parameter.

An implementation shall apply the specifications given in 8.1.1.

8.2.3.3 ResultingResourceSettings[ ] Array Parameter

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

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

8.2.3.4 Return Codes

An implementation shall indicate the result of the method execution by using the return code values specified in Table 7.

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

8.2.4 CIM_VirtualSystemManagementService.ModifyResourceSettings( ) Method (Conditional)

Behavior applicable to all extrinsic methods is specified in 8.1.2.

The implementation of the ModifyResourceSettings( ) method is conditional with respect to support of the modification of virtual systems and resources (see 7.4.6.3).

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 (Conditional)

Behavior applicable to all extrinsic methods is specified in 8.1.2.

The implementation of the ModifySystemSettings( ) method is conditional with respect to the implementation supporting the modification of virtual systems and resources (see 7.4.6.3).

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

Table 10 contains requirements for parameters of this method.

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.

8.2.5.1 SystemSettings Parameter

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

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

8.2.5.2 Return Codes

An implementation shall indicate the result of the method execution by using the return code values specified in Table 11.

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.

8.2.6 CIM_VirtualSystemManagementService.RemoveResourceSettings( ) Method (Conditional)

Behavior applicable to all extrinsic methods is specified in 8.1.2.

The implementation of the RemoveResourceSettings( ) method is conditional with respect to the support of the addition and the removal of virtual resources to virtual systems (see 7.4.6.2).

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 (Conditional)

Behavior applicable to all extrinsic methods is specified in 8.1.2.

The implementation of the CreateSnapshot( ) method is conditional with respect to the support of the creation, destruction and application of virtual system snapshots (see 7.7.1.1).

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.

Qualifiers	Name	Type	Description/Values
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

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 (Conditional)

Behavior applicable to all extrinsic methods is specified in 8.1.2.

The implementation of the DestroySnapshot( ) method is conditional with respect to the support of virtual system snapshots (see 7.7.1.1).

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.

1241 **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.

1242 ### 8.3.3 VirtualSystemSnapshotService.ApplySnapshot( ) Method (Conditional)

1243 Behavior applicable to all extrinsic methods is specified in 8.1.2.

1244 The implementation of the ApplySnapshot( ) method is conditional with respect to the support of virtual
1245 system snapshots (see 7.7.1.1).

1246 The execution of the ApplySnapshot( ) method shall indicate that the snapshot is used for the next
1247 activation of the associated virtual system (the virtual system that was the source for the snapshot). The
1248 method execution shall have one or both of the following effects:

1249 - The snapshot is persistently established to be used for subsequent activations.
1250 - The virtual system is immediately activated or recycled, using the snapshot.

1251 Table 18 contains requirements for parameters of this method.

1252 **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.

1253 #### 8.3.3.1 Snapshot Parameter

1254 A client shall set a value of the Snapshot parameter to refer to the instance of the
1255 CIM_VirtualSystemSettingData class that represents a snapshot.

1256 An implementation shall interpret the value of the Snapshot parameter to identify the snapshot that is to
1257 be applied.

1258 #### 8.3.3.2 Return Codes

1259 An implementation shall indicate the result of the method execution by using the return code values speci-
1260 fied in Table 19.

1261 **Table 19 – ApplySnapshot( ) Method: Return Code Values**

Value	Description

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

Support for operations for each profile class (including associations) is specified in the following subclauses. Each subclause includes either the statement "All operations in the default list in 8.4 are supported as described by DSP0200 version *1.2.0* or a table listing all of the operations that are not supported by this profile or where the profile requires behavior other than that described by DSP0200 version *1.2.0*.

The default list of operations is as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

A compliant implementation shall support all of the operations in the default list for each class, unless the "Requirement" column states something other than *Mandatory*.

This profile specification defines methods in terms of DSP0200 version *1.2.0*.

8.4.1 CIM_AffectedJobElement

Table 20 lists operations that either have special requirements beyond those from DSP0200 version *1.2.0* or shall not be supported.

Table 20 – Operations: CIM_AffectedJobElement Association

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

EnumerateInstances	Unspecified	None
EnumerateInstanceNames	Unspecified	None

8.4.2 CIM_ComputerSystem

All operations in the default list in 8.4 are supported as described by DSP0200 version *1.2.0*.

8.4.3 CIM_ConcreteJob

All operations in the default list in 8.4 are supported as described by DSP0200 version *1.2.0*.

8.4.4 CIM_Dependency

Table 22 lists operations that either have special requirements beyond those from DSP0200 version *1.2.0* or shall not be supported.

Table 21 – Operations: CIM_Dependency Association

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None
EnumerateInstances	Unspecified	None
EnumerateInstanceNames	Unspecified	None

8.4.5 CIM_ElementCapabilities

Table 22 lists operations that either have special requirements beyond those from DSP0200 version *1.2.0* or shall not be supported.

Table 22 – Operations: CIM_ElementCapabilities Association

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None
EnumerateInstances	Unspecified	None
EnumerateInstanceNames	Unspecified	None

8.4.6 CIM_ElementConformsToProfile

Table 23 lists operations that either have special requirements beyond those from DSP0200 version *1.2.0* or shall not be supported.

1298 **Table 23 – Operations: CIM_ElementConformsToProfile Association**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None
EnumerateInstances	Unspecified	None
EnumerateInstanceNames	Unspecified	None

1299 ### 8.4.7 CIM_HostedDependency

1300 Table 24 lists operations that either have special requirements beyond those from DSP0200 version *1.2.0*
1301 or shall not be supported.

1302 **Table 24 – Operations: CIM_HostedDependency Association**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None
EnumerateInstances	Unspecified	None
EnumerateInstanceNames	Unspecified	None

1303 ### 8.4.8 CIM_HostedService

1304 Table 25 lists operations that either have special requirements beyond those from DSP0200 version *1.2.0*
1305 or shall not be supported.

1306 **Table 25 – Operations: CIM_HostedService Association**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None
EnumerateInstances	Unspecified	None
EnumerateInstanceNames	Unspecified	None

1307 ### 8.4.9 CIM_LastAppliedSnapshot

1308 Table 26 lists operations that either have special requirements beyond those from DSP0200 version *1.2.0*
1309 or shall not be supported.

1310 **Table 26 – Operations: CIM_LastAppliedSnapshot Association**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None
EnumerateInstances	Unspecified	None
EnumerateInstanceNames	Unspecified	None

1311 ### 8.4.10 CIM_MostCurrentSnapshotInBranch

1312 Table 27 lists operations that either have special requirements beyond those from DSP0200 version *1.2.0*
1313 or shall not be supported.

1314 **Table 27 – Operations: CIM_MostCurrentSnapshotInBranch Association**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None
EnumerateInstances	Unspecified	None
EnumerateInstanceNames	Unspecified	None

1315 ### 8.4.11 CIM_ReferencedProfile

1316 Table 28 lists operations that either have special requirements beyond those from DSP0200 version *1.2.0*
1317 or shall not be supported.

1318 **Table 28 – Operations: CIM_ReferencedProfile Association**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None
EnumerateInstances	Unspecified	None
EnumerateInstanceNames	Unspecified	None

1319 ### 8.4.12 CIM_RegisteredProfile

1320 All operations in the default list in 8.4 are supported as described by DSP0200 version *1.2.0*.

8.4.13 CIM_ServiceAffectsElement

Table 29 lists operations that either have special requirements beyond those from DSP0200 version *1.2.0* or shall not be supported.

Table 29 – Operations: CIM_ServiceAffectsElement Association

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None
EnumerateInstances	Unspecified	None
EnumerateInstanceNames	Unspecified	None

8.4.14 CIM_SnapshotOfVirtualSystem

Table 30 lists operations that either have special requirements beyond those from DSP0200 version *1.2.0* or shall not be supported.

Table 30 – Operations: CIM_SnapshotOfVirtualSystem Association

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None
EnumerateInstances	Unspecified	None
EnumerateInstanceNames	Unspecified	None

8.4.15 CIM_System

All operations in the default list in 8.4 are supported as described by DSP0200 version *1.2.0*.

8.4.16 CIM_VirtualSystemManagementCapabilities

All operations in the default list in 8.4 are supported as described by DSP0200 version *1.2.0*.

8.4.17 CIM_VirtualSystemManagementService

All operations in the default list in 8.4 are supported as described by DSP0200 version *1.2.0*.

8.4.18 CIM_VirtualSystemSnapshotService

All operations in the default list in 8.4 are supported as described by DSP0200 version *1.2.0*.

8.4.19 CIM_VirtualSystemSnapshotCapabilities

All operations in the default list in 8.4 are supported as described by DSP0200 version *1.2.0*.

8.4.20 CIM_VirtualSystemSnapshotServiceCapabilities

All operations in the default list in 8.4 are supported as described by DSP0200 version *1.2.0*.

9 Use Cases (Informative)

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 CIM object manager (CIMOM) that exposes an implementation of this profile.
- The client is able to communicate with the CIMOM 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.

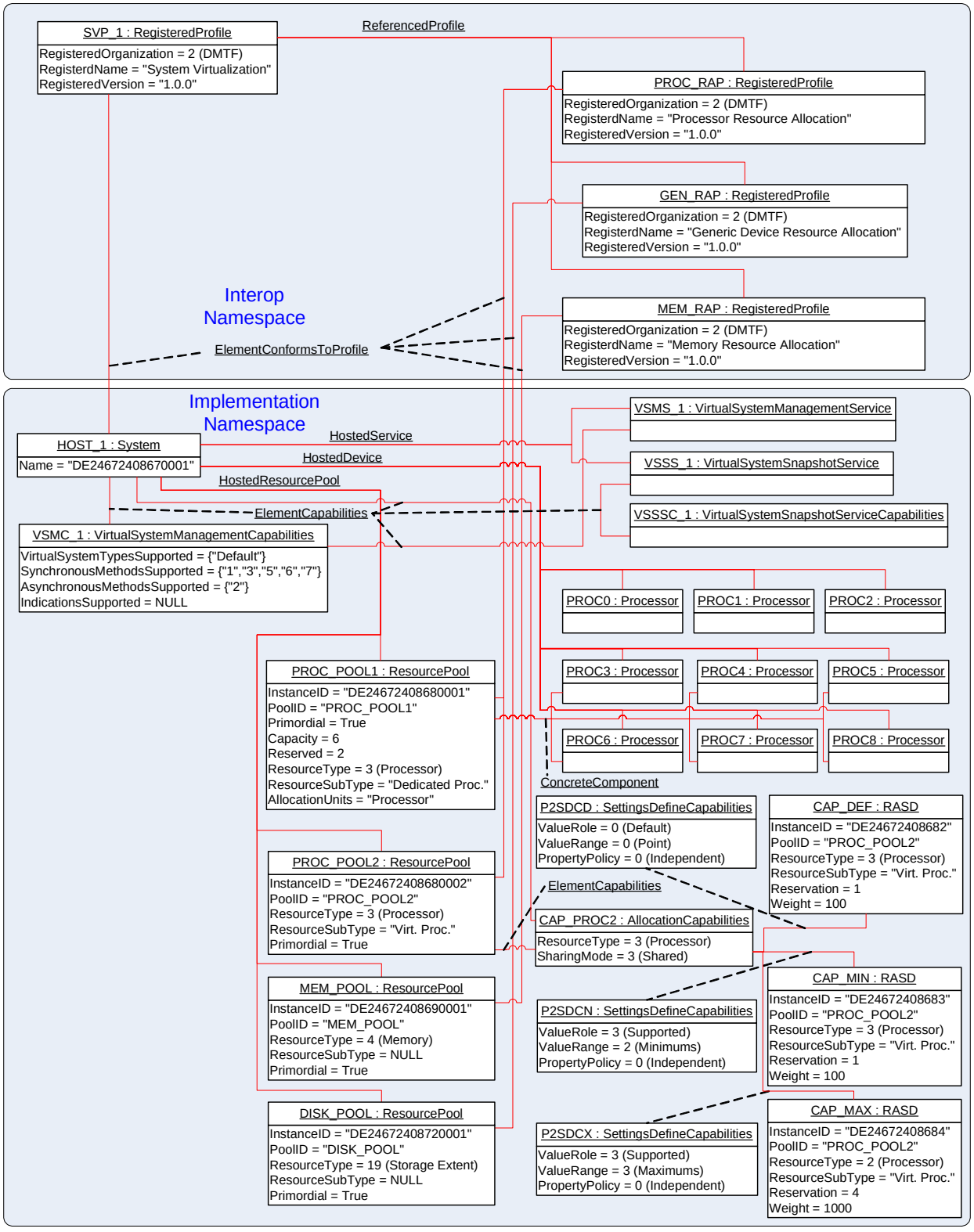


Figure 3 – System Virtualization Profile Instance Diagram: Discovery, Localization, and Inspection

9.2.1 SLP-Based Discovery of CIM Object Managers Hosting Implementations of This Profile

The service location protocol (SLP) is used to locate CIM object managers (CIMOMs). A CIMOM 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 CIMOM 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 CIMOM where this profile is implemented.

9.2.2 Locate Conformant Implementations Using the EnumerateInstances( ) Operation

Assumption: The client knows the URL of a CIMOM 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.

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

Result: The client knows a set of instances of the CIM_RegisteredProfile class, each representing an implementation of 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 CIMOM 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 That Are Central and Scoping 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 DMTF Management 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 the *System Virtualization 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 the *System Virtualization 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 the *System Virtualization Profile*. These instances represent implementations of scoped resource allocation DMTF management profiles:

- The instance tagged PROC_RAP represents an implementation of the *Processor Device Resource Virtualization Profile*.
- The instance tagged GEN_RAP represents an implementation of the *Generic Device Resource Virtualization Profile*.
- The instance tagged MEM_RAP represents an implementation of the *Memory Resource Virtualization Profile*.

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.
2) The client analyzes the instance of the CIM_VirtualSystemManagementCapabilities class.
   - The VirtualSystemTypesSupported[ ] array property lists identifiers of virtual system types that the implementation supports.
   - The SynchronousMethodsSupported[ ] array property lists identifiers of methods of the CIM_VirtualSystemManagementService class that are supported with synchronous method execution only.
   - The AsynchronousMethodsSupported[ ] array property lists identifiers of methods of the CIM_VirtualSystemManagementService class that are supported 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 supported by the implementation with synchronous execution.
- The AsynchronousMethodsSupported[ ] array property lists the enumerated value { 2 (DefineSystemSupported) }, which indicates that the DefineSystem( ) method is supported by the implementation with synchronous or asynchronous execution.
- The value of the IndicationsSupported[ ] array property is NULL, which indicates that indications are not supported 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.
2) For each instance of CIM_ResourcePool, the client determines whether the value of the ResourceType property matches the requested resource type.

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

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

9.2.9 Obtain a Set of Central Instances of Scoped Resource Allocation DMTF Management Profiles

Resource allocation DMTF management profiles are based on the abstract *Resource Allocation Profile* 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.

  The resulting list is the result for this use case.

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

9.2.10 Determine the Supported Resource Types of an Implementation

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 locates implementations of DMTF management profiles that are scoped by this profile (see 9.2.5).

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

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

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

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

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

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

- The instance tagged PROC_RAP represents an implementation of the *Processor Device Resource Virtualization Profile*.
- The instance tagged GEN_RAP represents an implementation of the *Generic Device Resource Virtualization Profile*.
- The instance tagged MEM_RAP represents an implementation of the *Memory Resource Virtualization Profile*.

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

- The instance tagged PROC_RAP is associated with two instances that represent resource pools for the allocation of processors. They show a value of 3 (Processor) for the ResourceType property and are tagged PROC_POOL1 and PROC_POOL2.
- The instance tagged GEN_RAP is associated with one instance that represents a resource pool for the allocation of virtual disks. It shows a value of 19 (Storage Extent) for the ResourceType property and is tagged DISK_POOL.
- The instance tagged MEM_RAP is associated with one instance that represents a resource pool for the allocation of memory. It shows a value of 4 (Memory) for the ResourceType property and is tagged MEM_POOL.

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

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.
2) The client drops instances from the result list of step 1) that have a value for the ResourceType property that does not match the requested resource type.

   The purpose of the following two steps is to further limit the result set from step 2) to those instances of the CIM_AllocationCapabilities class that describe default settings. Default settings are flagged in the connecting instance of the CIM_ElementCapabilities association that has a value of 2 (Default) for the Characteristics property.
3) For each instance of the list resulting from step 2), the client invokes the intrinsic References( ) CIM operation for a list of association instances that refer to the resource pool:
   – The value of the ObjectName parameter refers the instance of the CIM_ResourcePool class.
   – The value of the ResultClass parameter is set to “CIM_AllocationCapabilities”.

   The result is a list of instances of the CIM_ElementCapabilities association that associate an instance of the CIM_ResourcePool class that is taken from the result of step 2).
4) From the list obtained in step 3), the client drops all elements that meet either of the following conditions:
   – have a value other than 2 (Default) for the Characteristics property
   – do not refer to the instance of the CIM_System class that represents the host system through the ManagedElement property

   The list should now contain one instance of the CIM_AllocationCapabilities class that represents default allocation capabilities for the resource type in question.

5) The client invokes the intrinsic Associators( ) CIM operation to resolve association for the resource pool, as follows:

   – The value of the ObjectName parameter refers to the instance of the CIM_AllocationCapabilities class selected in step 4).

   – The value of the AssocClass parameter is set to “CIM_ElementCapabilities”.

   – The value of the ResultClass parameter is set to “CIM_ResourcePool”.

   The result is a list of instances of the CIM_ResourcePool class. The size of the list is 1.

Result: The client knows the instance of the CIM_ResourcePool class that represents the default resource pool for the requested resource type.

In the example shown in Figure 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 the *Allocation Capabilities Profile* shows only one associated element for each resource type.
- With step 2), the client reduces the result set to the one element that describes allocation capabilities processors. This instance is tagged CAP_PROC1.
- With steps 3) and 4), the client further reduces the result set to the one instance of the CIM_AllocationCapabilities class that represents the system’s default capabilities for resource type 3 (Processor).
- With step 5), the client resolves the CIM_ElementCapabilities association in order to obtain the instance of the CIM_ResourcePool class that represents the default resource pool for processors. This instance is tagged PROC_POOL2.

9.2.12 Determine Resource Pool for a Resource Allocation Request or 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.
2) The client drops from the result of step 1) those instances in which the ResourceType property designates a resource type other than the requested resource type. This step is required only if the starting point of the use case was an instance of the CIM_System class.

   At this point the client has a list of instances of the CIM_AllocationCapabilities class that describe allocation capabilities. The value of the SharingMode property allows a distinction between shared and dedicated resources.
3) The client invokes the intrinsic References( ) CIM operation for a set of instances of the CIM_SettingsDefineCapabilities association that each associate one instance of the CIM_ResourceAllocationSettingData class that describes a limiting aspect (min/max/increment), as follows:
   – The value of the ObjectName parameter is set to refer to the instance of the CIM_AllocationCapabilities class.
   – The value of the ResultClass parameter is set to “CIM_SettingsDefineCapabilities”.

   The result is a list of instances of the CIM_SettingsDefineCapabilities association.
4) For each instance that is a result from step 3), the client analyzes the values of the PropertyPolicy property and the ValueRange property. The value of the ValueRole property is irrelevant in this case.

   The property values have the following impact:
   – The value of the PropertyPolicy property is 0 (Independent) for a conformant implementation of the *Allocation Capabilities Profile* in association instances that connect a min/max/increment limiting setting.
   – The value of the ValueRange property allows determining the designation of the associated setting:
     – A value of 1 (Minimums) indicates that the referenced instance of the CIM_ResourceAllocationSettingData class represents a lower limit for the allocation of resources of the respective resource type.
     – A value of 2 (Maximums) indicates that the referenced instance of the CIM_ResourceAllocationSettingData class represents an upper limit for the allocation of resources of the respective resource type.
     – A value of 3 (Increments) indicates that the referenced instance of the CIM_ResourceAllocationSettingData class represents an increment for the allocation of resources of the respective resource type.
5) For each association instance obtained in step 4), the client invokes the intrinsic GetInstance( ) CIM operation for the instance of the CIM_ResourceAllocationSettingData class that describes

the respective limitation. The value of InstanceName parameter is set to the value of the PartComponent property in the association instance obtained in step 4).

In each case, the result is an instance of the CIM_ResourceAllocationSettingData class that represents a limiting setting.

Result: The client knows the valid resource settings for the requested resource type.

9.2.14 Determine Implementation Class Specifics

The *System Virtualization 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.
2) The client invokes the intrinsic GetClass( ) CIM operation to obtain a formal class description, as follows:
   – The value of the ClassName parameter is set to the name of the class.
   – The value of the LocalOnly parameter is set to “false”.
   – The value of the IncludeQualifiers parameter is set to “true”.
   – The value of the IncludeClassOrigin parameter is set to “true”.

   The result is a description of a CIM class.

Result: The client has a description of the class. The format depends on the CIM client used to issue the request and is based on the XML class data structure that describes a CIM class as defined in the *Specification for the Representation of CIM in XML*, version *2.2.0*. 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”.
2) 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

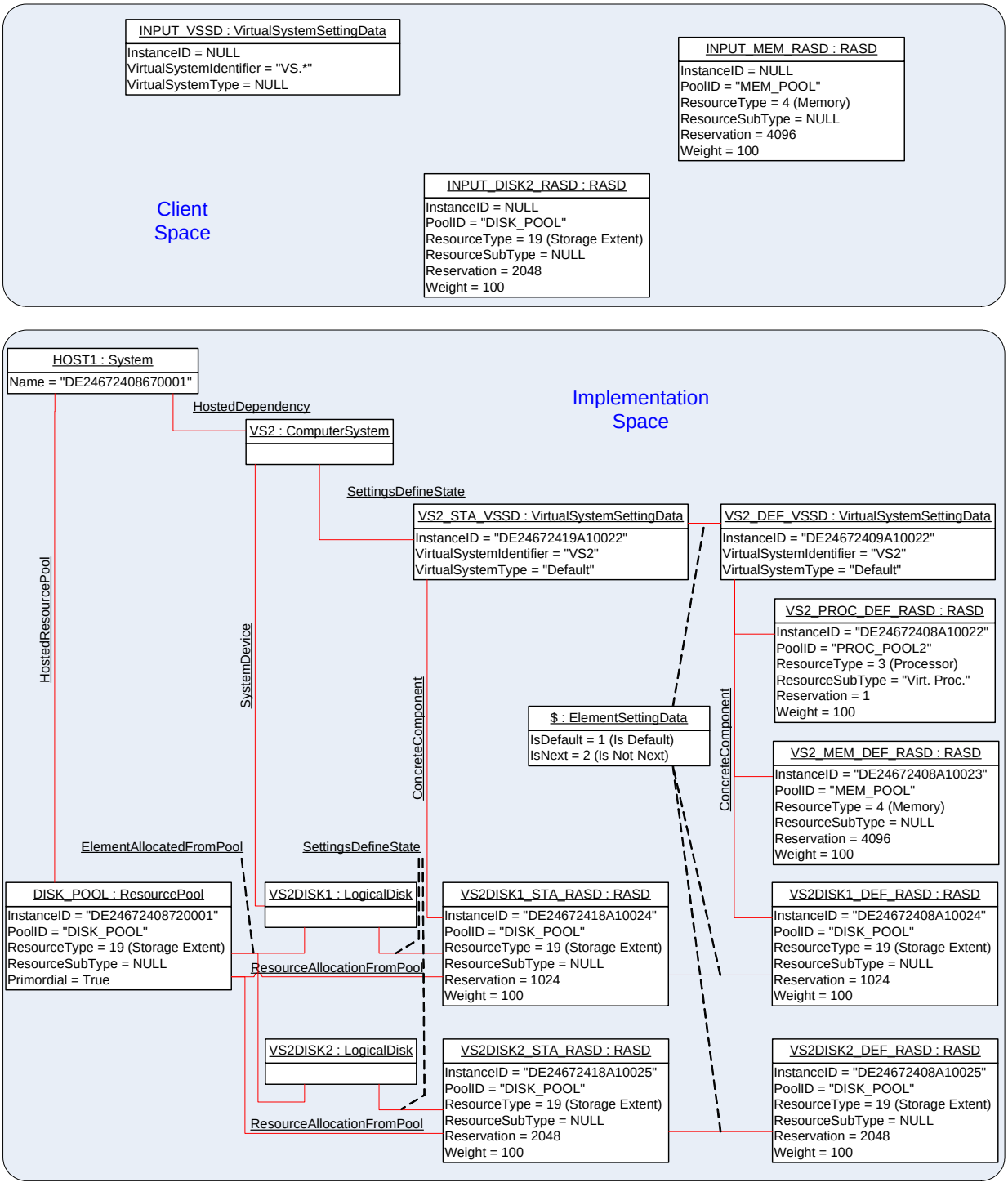
Assumption: No assumption is made beyond the general assumption specified in 9.3.

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.
2) The implementation executes the DefineSystem( ) method. The configuration of the new virtual system is created according to the client’s requirements. The new virtual system is in the “Defined” virtual system state.

   The value returned in the ResultingSystem parameter refers to an instance of the CIM_ComputerSystem class.

Result: The client knows a reference to an instance of the CIM_ComputerSystem class that represents the new virtual system.

1778 Figure 4 shows the representation of a virtual system that was defined using an “Input” virtual system and
1779 a “Reference” virtual system configuration.



1780

1781 **Figure 4 – Virtual System Configuration Based on "Input" Virtual System Configurations and**
1782 **Implementation Defaults**

1783 The new virtual system is represented by an instance of the CIM_ComputerSystem class that is tagged
1784 VS2. The right side of Figure 4 shows the “Defined” virtual system configuration for the new virtual sys-

tem. 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 the *Virtual System Profile*).
2) The client makes conformant changes to the instance of the CIM_VirtualSystemSettingData class. In particular, the client must not modify key properties.
3) The client invokes the ModifySystemSettings( ) method (see 8.2.5) on the virtual system management service. The value of the SystemSettings parameter is the modified instance from step 2).
4) The implementation executes the ModifySystemSettings( ) method, and the configuration of the virtual system is modified according to the clients requirements.

Result: The requested modification is applied to the state or definition of the virtual system.

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.
2) 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).
3) The implementation executes the AddResourceSettings( ) method, adding the requested resource allocations and resource allocation requests to the virtual system configuration.

Result: The requested resource allocations or resource allocation requests are configured into the referenced virtual system configuration.

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 4).

1) The client invokes the intrinsic Associators( ) CIM operation for the virtual resource state extension as follows:
   – The value of the ObjectName parameter is set to refer to the instance of the CIM_LogicalDevice class.
   – The value of the AssocClass parameter is set to “CIM_SettingsDefineState”.
   – The value of the ResultClass parameter is set to “CIM_ResourceAllocationSettingData”.

   The result is a list of instances of the CIM_ResourceAllocationSettingData class. The size of the list is expected to be 1, and that element represents the virtual resource state extension. If the client intends to modify the virtual resource state extension, the client skips steps 2) and 3), and proceeds with step 4). If the client intends to modify the virtual resource definition, the client continues with step 2).
2) The client invokes the intrinsic References( ) CIM operation for the association instances that connect the virtual resource definition, as follows:
   – The value of the ObjectName parameter is set to refer to the instance of the CIM_ResourceAllocationSettingData class that was obtained in step 1).
   – The value of the ResultClass parameter is set to “CIM_ElementSettingData”.

   The result is a list of instances of the CIM_ElementSettingData association that connect various settings to the virtual resource state extension.
3) The client selects from the result set of step 2) the instance in which the IsDefault property has a value of 1 (Is Default). In that instance, the value of the SettingData property refers to the in-

stance of the CIM_ResourceAllocationSettingData class that represents the virtual resource definition.

4) The client invokes the intrinsic GetInstance( ) CIM operation for the setting that represents the resource allocation definition. The value of the InstanceName parameter is set to the value of the SettingData property from the instance of the CIM_ElementSettingData association selected in step 3).

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

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

   Eventually the client executes steps 1) to 5) repetitively, preparing a set of resource allocation change requests that subsequently are applied as one atomic operation.

6) The client invokes the ModifyResourceSettings( ) method (see 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 5).

7) The implementation executes the ModifyResourceSettings( ) method, causing the requested resource allocation changes being applied to resource allocation state extensions or resource allocation definitions.

Result: The requested resource modifications are applied to virtual resource state extensions or virtual resource definitions.

Figure 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.

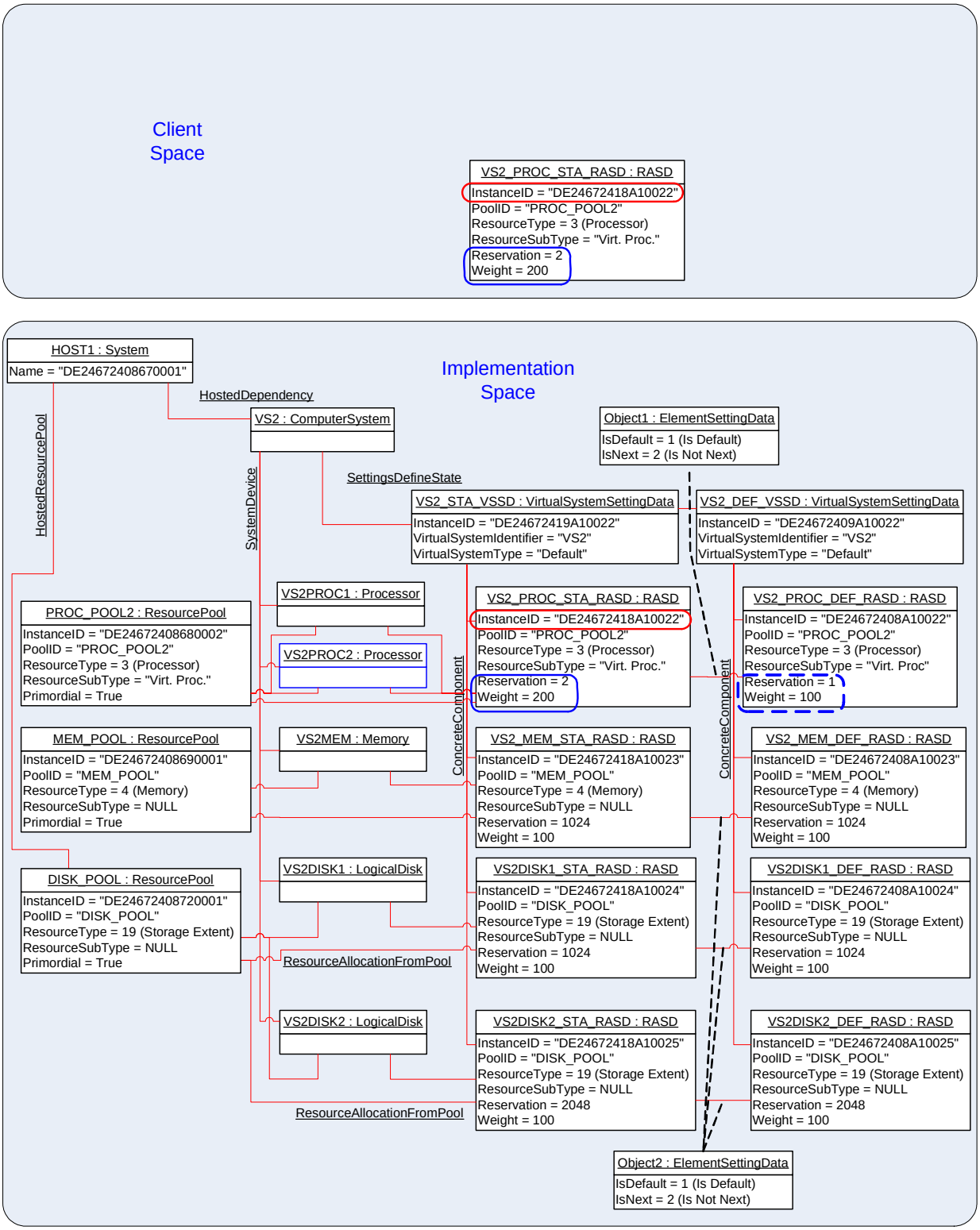


Figure 5 – Virtual System Resource Modification

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 Clause 9 of the *Virtual System Profile* 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

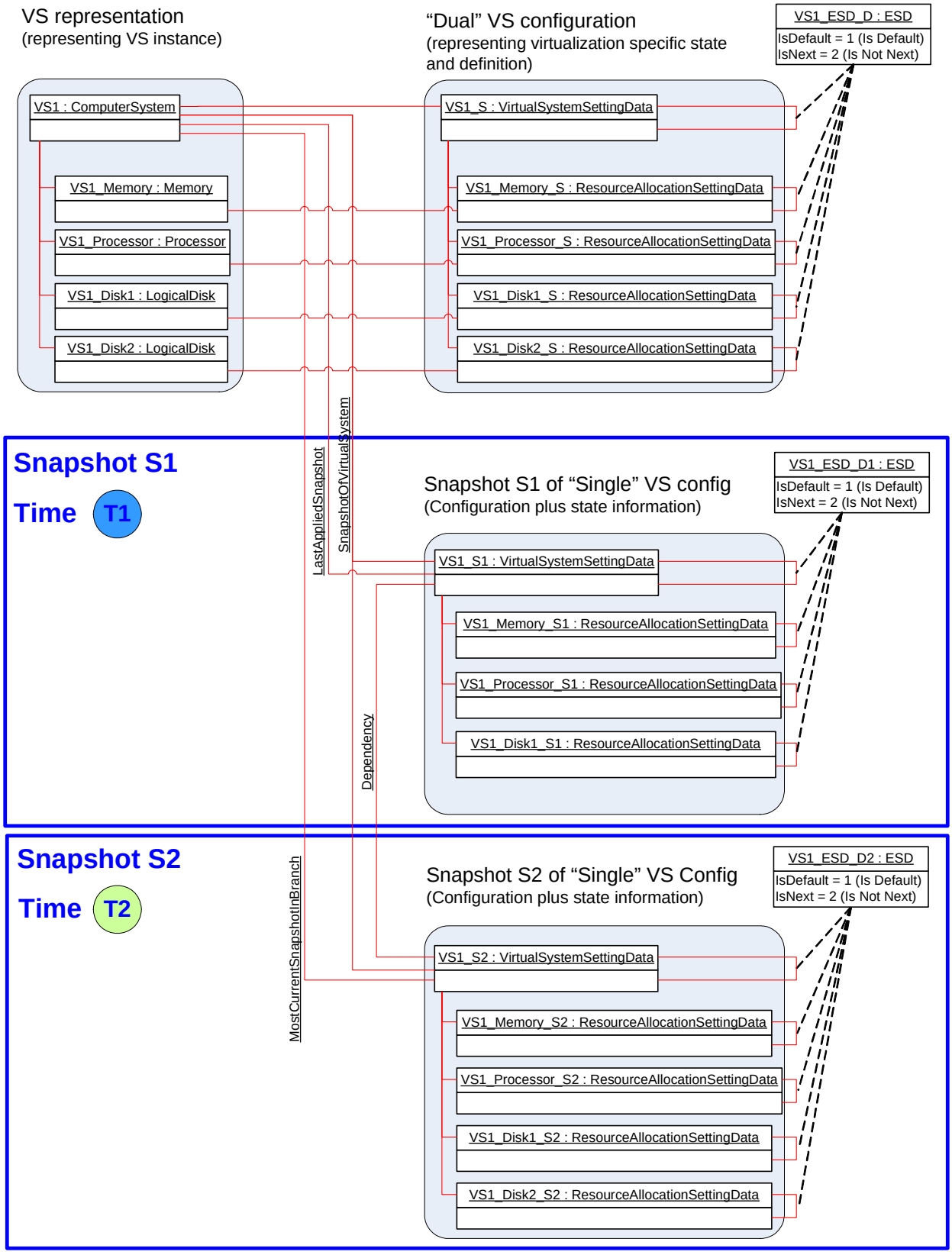
Figure 6 depicts the CIM representation of a virtual system VS1 and of configurations that are associated with the virtual system at time T3. In the example, it is assumed that the implementation applies the "Single-Configuration Implementation Approach" as described in the *Virtual System Profile*.

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.
2) At a time after T0 but before T1, the virtual system is activated.
3) At time T1, a full snapshot S1 is captured of the virtual system. Virtual system definition and state are copied into the snapshot. A full snapshot includes the "content" of virtual memory *and* of virtual disks; a disk snapshot would contain the "content" of virtual disks only.
4) The virtual system remains active after the snapshot is captured. The virtual system configuration and the "content" of memory and of virtual disks may change in that interval.
5) At a time after T1 but before T2, snapshot S1 is applied to the virtual system, causing definition and state to be restored to the situation at time T1.
6) Still at a time before T2, a second virtual disk is dynamically added to the virtual system. Because in this example the implementation applies the "Single-Configuration Implementation Approach," this change in effect applies to both virtual system definition and virtual system instance and is visible through the "Single" VS configuration.
7) At time T2, snapshot S2 is captured of the virtual system. Because at time T2 the virtual system snapshot S1 is the last applied snapshot, snapshot S2 depends on snapshot S1.
8) The virtual system remains active after the snapshot is captured. The virtual system configuration and the "content" of memory and of virtual disks may change in that interval.
9) At a time after T2 but before T3, snapshot S2 is applied to the virtual system, causing definition and state to be restored to the situation at time T2, thereby nullifying changes that were applied to the virtual system after T2.
10) At time T3, the situation is as shown in Figure 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



1978

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.
2) The client analyzes the instance of the CIM_VirtualSystemSnapshotServiceCapabilities class.
   - The SynchronousMethodsSupported[ ] array property lists identifiers of methods of the CIM_VirtualSystemSnapshotServiceCapabilities class that are supported with synchronous method execution only.
   - The AsynchronousMethodsSupported[ ] array property lists identifiers of methods of the CIM_VirtualSystemSnapshotServiceCapabilities class that are supported 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).
2) The implementation executes the CreateSnapshot( ) method.

   The value returned in the ResultingSnapshot parameter refers to an instance of the CIM_VirtualSystemSettingData class that represents the new snapshot.

Result: The client knows a reference to the instance of the CIM_VirtualSystemSettingData class that represents the created virtual system snapshot.

In the example shown in Figure 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 31 lists CIM elements that are defined or specialized for this profile. Each CIM element shall be implemented as described in Table 31. 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 31 – 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.

Element Name	Requirement	Description
CIM_VirtualSystemSettingData (Input)	Conditional	See 10.22.
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 (Conditional)

The implementation of the CIM_AffectedJobElement association is conditional with respect to the support of a non-NULL value for at least one element of the AsynchronousMethodsSupported[ ] array property of the CIM_VirtualSystemManagementCapabilities class.

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 32 contains the requirements for elements of this association.

Table 32 – 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 (Conditional)

The implementation of the CIM_ConcreteJob class is conditional with respect to the support of a non-NULL value for at least one element of the AsynchronousMethodsSupported[ ] array property of the CIM_VirtualSystemManagementCapabilities class.

An implementation shall use an instance of the CIM_ConcreteJob class to represent an asynchronous task.

Table 33 contains requirements for elements of this class.

Table 33 – 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 (Conditional)

The implementation of the CIM_Dependency association is conditional with respect to the support of virtual system snapshots (see 7.7.1.1).

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 34 contains requirements for elements of this class.

Table 34 – 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 35 contains requirements for elements of this association.

Table 35 – 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) (Conditional)

The implementation of the CIM_ElementCapabilities association for the virtual system management service is conditional with respect to the support of any of:

- 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)

2192 An implementation shall use an instance of the CIM_ElementCapabilities association to associate an in-
2193 stance of the CIM_VirtualSystemManagementService class that represents a virtual system management
2194 service with an instance of the CIM_VirtualSystemManagementCapabilities that describes the capabilities
2195 of the virtual system management service.

2196 Table 36 contains requirements for elements of this association.

2197 **Table 36 – 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

2198 10.6 CIM_ElementCapabilities (Virtual System Snapshot Service) (Conditional)

2199 The implementation of the CIM_ElementCapabilities association for the virtual system snapshot service is
2200 conditional with respect to the support of virtual system snapshots (see 7.7.1.1).

2201 An implementation shall use an instance of the CIM_ElementCapabilities association to associate an in-
2202 stance of the CIM_VirtualSystemSnapshotService class that represents a virtual system snapshot service
2203 with an instance of the CIM_VirtualSystemSnapshotServiceCapabilities class that describes the capabili-
2204 ties of the virtual system snapshot service.

2205 Table 37 contains requirements for elements of this association.

2206 **Table 37 – 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

2207 10.7 CIM_ElementCapabilities (Snapshots of Virtual Systems) (Conditional)

2208 The implementation of the CIM_ElementCapabilities association for the virtual systems snapshots is
2209 conditional with respect to the support of virtual system snapshots (see 7.7.1.1).

2210 The implementation shall use an instance of the CIM_ElementCapabilities association to associate in-
2211 stances of the CIM_VirtualSystemSnapshotCapabilities class with those instances of the
2212 CIM_ComputerSystem class that represent a virtual system to which the capabilities apply.

2213 Table 38 contains requirements for elements of this association.

2214 **Table 38 – 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

2215 ## 10.8 CIM_ElementConformsToProfile

2216 An implementation shall use an instance of the CIM_ElementConformsToProfile association to associate
2217 an instance of the CIM_RegisteredProfile class that represents an implementation of this profile with
2218 instances of the CIM_System class that represent a host system that is a central and scoping instance of
2219 this profile.

2220 Table 39 contains requirements for elements of this association.

2221 **Table 39 – 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:** *

2222 ## 10.9 CIM_HostedDependency

2223 An implementation shall use an instance of the CIM_HostedDependency association to associate an
2224 instance of the CIM_System class that represents a host system with each instance of the CIM_Comput-
2225 erSystem class that represents a virtual system hosted by the host system.

2226 Table 40 contains requirements for elements of this association.

2227 **Table 40 – 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) (Conditional)

The implementation of the CIM_HostedService association for the virtual system management service is conditional with respect to the support of any of:

- 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)

The implementation shall use an instance of the CIM_HostedService association to associate an instance of the CIM_System class that represents a host system and the instance of the CIM_VirtualSystem-ManagementService class that represents the virtual system management service that is hosted by a host system.

Table 41 contains requirements for elements of this association.

Table 41 – 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) (Conditional)

The implementation of the CIM_HostedService association is conditional with respect to the support of virtual system snapshots (see 7.7.1.1).

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 42 contains requirements for elements of this association.

Table 42 – 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 (Conditional)

The implementation of the CIM_LastAppliedSnapshot association is conditional with respect to the support of virtual system snapshots (see 7.7.1.1).

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 43 contains requirements for elements of this association.

Table 43 – 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 (Conditional)

The implementation of the CIM_MostCurrentSnapshotInBranch association is conditional with respect to the support of virtual system snapshots (see 7.7.1.1).

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 44 contains requirements for elements of this association.

Table 44 – 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 (Conditional)

The implementation of the CIM_ReferencedProfile association is conditional with respect to the support of resource allocation DMTF management profiles like the *Generic Device Resource Virtualization Profile*.

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 45 contains requirements for elements of this association.

Table 45 – 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 46 contains requirements for elements of this class.

Table 46 – 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) (Conditional)

The implementation of the CIM_ServiceAffectsElement association for the virtual system management service is conditional with respect to the support of any of:

- 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)

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 manage-

ment service and any instance of the CIM_ComputerSystem class that represents a virtual system that is managed by that virtual system management service.

Table 47 contains requirements for elements of this association.

Table 47 – 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) (Conditional)

The implementation of the CIM_ServiceAffectsElement association is conditional with respect to the support of virtual system snapshots (see 7.7.1.1).

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 48 contains requirements for elements of this association.

Table 48 – 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 (Conditional)

The implementation of the CIM_SnapshotOfVirtualSystem association is conditional with respect to the support of virtual system snapshots (see 7.7.1.1).

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

2315 for the virtual system snapshot and the instance of the CIM_VirtualSystemSettingData class that repre-
2316 sents a snapshot of the virtual system

2317 Table 49 contains requirements for elements of this association.

2318 **Table 49 – Association: CIM_SnapshotOfVirtualSystem**

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Reference to the instance of the CIM_Computer-System class that represents the source virtual system **Cardinality:** 0..1
Dependent	Mandatory	**Key:** Reference to an instance of the CIM_VirtualSys-temSettingData class that represents a virtual system snapshot **Cardinality:** *

2319 ## 10.19 CIM_System

2320 An implementation shall use an instance of the CIM_System class to represent a host system.

2321 Table 51 contains requirements for elements of this class.

2322 **Table 50 – Class: CIM_VirtualSystemManagementCapabilities**

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

2323 ## 10.20 CIM_VirtualSystemManagementCapabilities

2324 An implementation shall use an instance of the CIM_VirtualSystemManagementCapabilities class to
2325 represent the virtual system management capabilities of a host system.

2326 Table 51 contains requirements for elements of this class.

2327 **Table 51 – 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.

2328 ## 10.21 CIM_VirtualSystemManagementService (Conditional)

2329 The implementation of the CIM_VirtualSystemManagementService class is conditional with respect to the
2330 support of any of:

2331 - 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)

An implementation shall use an instance of the CIM_VirtualSystemManagementService class to represent the virtual system management service provided by one host system.

Table 52 contains requirements for elements of this class.

Table 52 – Class: CIM_VirtualSystemManagementService

Elements	Requirement	Notes
CreationClassName	Mandatory	**Key**
Name	Mandatory	**Key**
SystemCreationClassName	Mandatory	**Key**
SystemName	Mandatory	**Key**
ApplySnapshot( )	Conditional	See 8.3.3.
AddResourceSettings( )	Conditional	See 8.2.3.
CreateSnapshot( )	Conditional	See 8.3.1.
DefineSystem( )	Conditional	See 8.2.1.
DestroySnapshot( )	Conditional	See 8.3.2.
DestroySystem( )	Conditional	See 8.2.2.
ModifyResourceSettings( )	Conditional	See 8.2.4.
ModifySystemSettings( )	Conditional	See 8.2.5.
RemoveResourceSettings( )	Conditional	See 8.2.6.

10.22 CIM_VirtualSystemSettingData (Input) (Conditional)

The implementation of the CIM_VirtualSystemSettingData class for input is conditional with respect to the support of any of:

- 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)

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

Table 53 contains requirements for elements of this class.

Table 53 – Class: CIM_VirtualSystemSettingData (Input)

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

10.23 CIM_VirtualSystemSettingData (Snapshot) (Conditional)

The implementation of the CIM_VirtualSystemSettingData class for the representation of snapshots of virtual systems is conditional with respect to the support of virtual system snapshots (see 7.7.1.1).

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

Table 54 contains requirements for elements of this class.

Table 54 – 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.		

10.24 CIM_VirtualSystemSnapshotCapabilities (Optional)

The implementation of the optional CIM_VirtualSystemSnapshotCapabilities class is specified only if virtual system snapshots are supported; (see 7.7.1.1).

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

2360 Table 55 contains requirements for elements of this class.

2361 **Table 55 – Class: CIM_VirtualSystemSnapshotCapabilities**

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

2362 10.25 CIM_VirtualSystemSnapshotService (Optional)

2363 The implementation of the CIM_VirtualSystemSnapshotService class is optional; if implemented it
2364 indicates the presence of the support of virtual system snapshots (see 7.7.1.1).

2365 An instance of the CIM_VirtualSystemSnapshotService class shall be used to represent the virtual system
2366 snapshot service available at a host system.

2367 Table 56 contains requirements for elements of this class.

2368 **Table 56 – 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.

2369 10.26 CIM_VirtualSystemSnapshotServiceCapabilities (Conditional)

2370 The implementation of the CIM_VirtualSystemSnapshotServiceCapabilities class is conditional with re-
2371 spect to the support of virtual system snapshots (see 7.7.1.1).

2372 An instance of the CIM_VirtualSystemSnapshotServiceCapabilities class shall be used to represent the
2373 capabilities of a virtual system snapshot service.

2374 Table 57 contains requirements for elements of this class.

2375 **Table 57 – Class: CIM_VirtualSystemSnapshotServiceCapabilities**

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

2376

ANNEX A
(Informative)

Change Log

Version	Date	Description
1.0.0a	2007/08/03	Initial creation

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