
# CIM System Virtualization Model White Paper

**Version 1.0.0**
**Status: Informational**
**Publication Date: 11/11/2007 6:31 AM**
**DSP2013**

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

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patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a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

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

# CIM System Virtualization Model White Paper
# CIM Version 2.16
# Version 1.0.0
# Publication Date: 11/11/2007 6:31 AM
# DSP2013
# Status: Informational


## Abstract

The DMTF Common Information Model (CIM) is a conceptual information model for describing computing and business entities in Internet, enterprise, and service-provider environments. CIM uses object-oriented techniques to provide a consistent definition of and structure for data. The CIM Schema establishes a common conceptual framework that describes the managed environment.

This white paper describes the CIM model for system virtualization, including the schema additions for the general resource allocation pattern and the modeling of virtual and host computer systems. The target audience of this white paper is anyone who wants to understand the modeling of system virtualization using CIM. Some familiarity with virtualization and the general concepts of the CIM model is assumed.



## Acknowledgments

The author acknowledges the contributions from the members of the DMTF System Virtualization, Partitioning, and Clustering Work Group.

# Table of Contents



# List of Figures

# 1 Introduction

This white paper describes the CIM model for system virtualization, including the schema additions for the general resource allocation pattern and the modeling of virtual and host computer systems. The model elements described in this paper enable management of system virtualization environments including management of virtual computer systems and their associated virtual resources and host computer system virtualization including resource pools and allocation from those pools.

## 1.1 Background Reference Material

This section lists approved references and references that are currently under development.

### 1.1.1 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 DSP1026, *System Memory 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 DSP1042, *System Virtualization Profile 1.0*

DMTF DSP1043, *Allocation Capabilities Profile 1.0*

DMTF DSP1052, *Computer System Profile 1.0*

DMTF DSP1057, *Virtual System Profile 1.0*

DMTF DSP1059, *Generic Device Resource Virtualization Profile 1.0*

SNIA, *Storage Management Initiative Specification (SMI-S)*

### 1.1.2 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, *File Based Storage Resource Virtualization Profile*

DMTF DSP1049, *Storage Adapter Resource Virtualization Profile 0.7*

## 1.2 Terminology

| Term | Definition |
|---|---|
| allocated resource | The partitioned or virtual resource that has been allocated to a consumer based on the associated resource allocation |
| child pool | A pool whose resources are backed by other resource pools. A child pool is a consumer of resources from its parent resource pools. All child pools contain no host resources; instead, they draw their resources from their parent pools through resource allocations. |
| consumer | The entity that receives allocated resources, for example, a virtual system or a child resource pool |
| current resource allocation setting data | The resource allocation setting data associated with the current allocation state of an allocated resource. These settings may differ from the defined resource allocation setting data if the host system supports the dynamic modification of a resource allocation. |
| dedicated virtual resource | A virtual resource that has been given exclusive use of a host resources. The host resources is not shared with any other consumer. |
| defined resource allocation setting data | The data associated with an allocated resource that describes the allocation settings to be used when that allocated resource is exposed to a virtual system during its instantiation or re-instantiation. |
| device resource allocation | The resource allocation to a consumer where there is a logical device representing the resource allocated. |
| host resource | A device or computing resource contained by the host system that may be allocated with either exclusive or shared access through the host system to provide resources to a resource pool or consumer |
| host system | A system that contains resources that may be allocated or virtualized |
| pass-through resource allocation | A resource allocation to a consumer in which the virtual resource is logically identical to the allocated host resource |
| resource allocation | The definition of the resource allocated to a consumer. It may be used to instantiate virtual resources. |
| resource allocation setting data (RASD) | Settings that define the resource allocation. These settings are used by the host system to manage the allocated resource and its relationship to the host resources and/or the resource pool from which it was allocated. |

| | |
|---|---|
| defined RASD | The RASD data representing the resource allocation request related to a currently not allocated resource. It decribes the allocation settings to be used when that resource is allocated to a virtual system during its (re)instantiation. |
| current RASD | The RASD representing the resource allocation of a currently allocated resource. These settings may differ from the defined RASD if the host system supports the dynamic modification of a resource allocation. |
| resource pool | An abstract entity used by the host system for the purpose of allocating and exposing allocated resources to consumers |
| resource type | A generic type that categorizes classes of resources (for example, Processor, Memory, Network Adapter, and so on) |
| shared virtual resource | An allocated resource that has been given the use of host resources that may also be shared with other consumers |
| simple resource allocation | The resource allocation to a consumer in which there is no logical device representing the resource allocated |
| virtual computer system | A virtual system as applied to a computer system. Other common industry terms for such a system include: Virtual Machine, Hosted Computer, Child Partition, Logical Partition, Domain, Guest, and Container. |
| virtual resource | The instantiation of the allocated resource that is exposed to a consumer through a logical device |
| virtual system | A system that is composed of allocated resources that may be partitioned or virtualized resources |
| virtual system setting data (VSSD) | Settings that define virtual system configuration data. |

# 2 Overview

The CIM system virtualization model, including CIM schema additions and a set of supporting profile documents, enables the management of system virtualization. Virtualization is a substitution process producing virtual resources which change aspects of the way consumers interact with the resources. These virtual resources are usually based on underlying physical resources, but they may have different properties or qualities. For example, virtual resources may have different capacities or sizes than the underlying resources or may have different qualities of service, such as improved performance or reliability. In system virtualization a host computer system provides the underlying resources that compose virtual computer systems and their constituent virtual devices.

## 2.1 Requirements

The following general requirements were considered during the design of the system virtualization model:

- Enable clients that are unaware of virtualization to manage virtual systems. That is, after a virtual computer system is created, most management operations (such as list, install, configure, show devices) should be available similarly on virtual or physical systems.
- The model should be flexible and general enough to support all types of platform virtualization including hypervisor-based virtualization, logical and physical partitioning, and operating system containers. The general patterns developed to model resource virtualization should be applicable as new types of virtualization become available.
- Because the capabilities of system virtualization implementations vary widely, the model should support the runtime inspection of a system's capabilities so that a client does not need *a priori* knowledge about an implementation's capabilities for the system to be managed effectively. This includes the ability to determine supported resource types, resources, and lifecycle capabilities.
- Management operations should be modeled such that reasonable defaults are made available wherever possible.
- The model should be extensible, with clear mechanisms for adding implementation-specific capabilities and for allowing a client to discover these capabilities.
- The model should leverage existing work that the DMTF (Server Management Work Group and Desktop & Mobile Work Group) has done for computer systems and their associated devices, and that SNIA has done for storage related modeling (see the *SMI-S*).

The following requirements relate to modeling of virtual and host computer systems and their associated resources:

- The model should support the capability to discover and enumerate virtual computer systems, host computer systems, and the relationships between them.

- The model should support the capability to create virtual computer systems by specifying resources (such as CPU, memory, network, and disk) and attributes (shared, virtualized, based on specific resource, and so on) for those resources. Deletion and modification of virtual computer systems should be supported.
- The model should support creation, deletion, modification and inventory of virtual resources.
- The model should support the ability, where feasible, to determine the mapping of virtual resources to the underlying host resource through as many layers of virtualization as required. For example, a customer that is notified that a particular physical disk is receiving intermittent errors should be able to determine which virtual machines would be affected if the disk failed. This may require combining information from multiple modeling domains.

## 2.2 Basic Virtual Computer System Modeling

The basic elements of a system virtualization environment are shown in Figure 1. The resources that make up the virtualization environment typically are supplied by one or more host computer systems. A virtualization layer (usually firmware or software, but possibly hardware) manages the lifecycle of a virtual computer system, which is composed of resources allocated or assigned from the host computer system. A virtual computer system may be active and running an operating system and applications with a full complement of virtual devices defined and allocated, or it may be inactive with no software running and a subset of the virtual devices actually allocated.

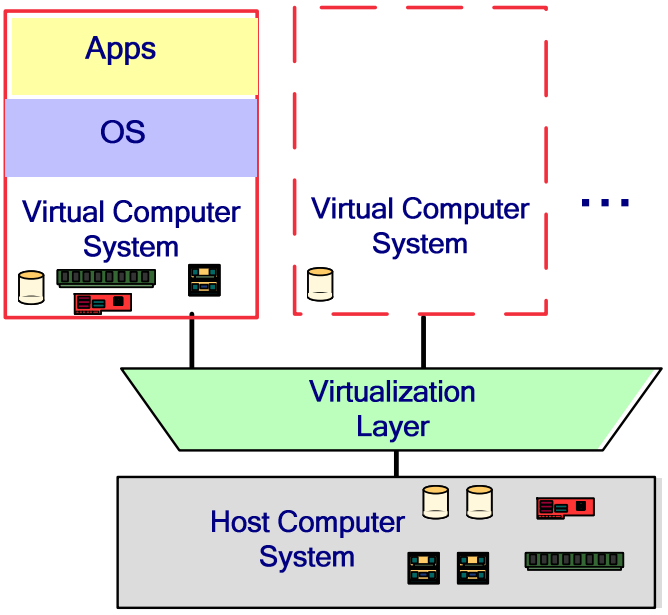


**Figure 1 Elements of the System Virtualization Environment**

The system virtualization model enables the client to manage the virtualization layer and the full lifecycle of the hosted virtual computer systems.

The basic elements of the system virtualization model are shown in Figure 2. Both host and virtual computer systems (also known in the industry as a virtual machine) are represented similarly by instances of the CIM_ComputerSystem class. Computer system devices are modeled through instances of subclasses of the CIM_LogicalDevice class. The relationship between system and devices is modeled through the CIM_SystemDevice association. The relationship of virtual computer systems to their host system is modeled through the CIM_HostedDependency association.

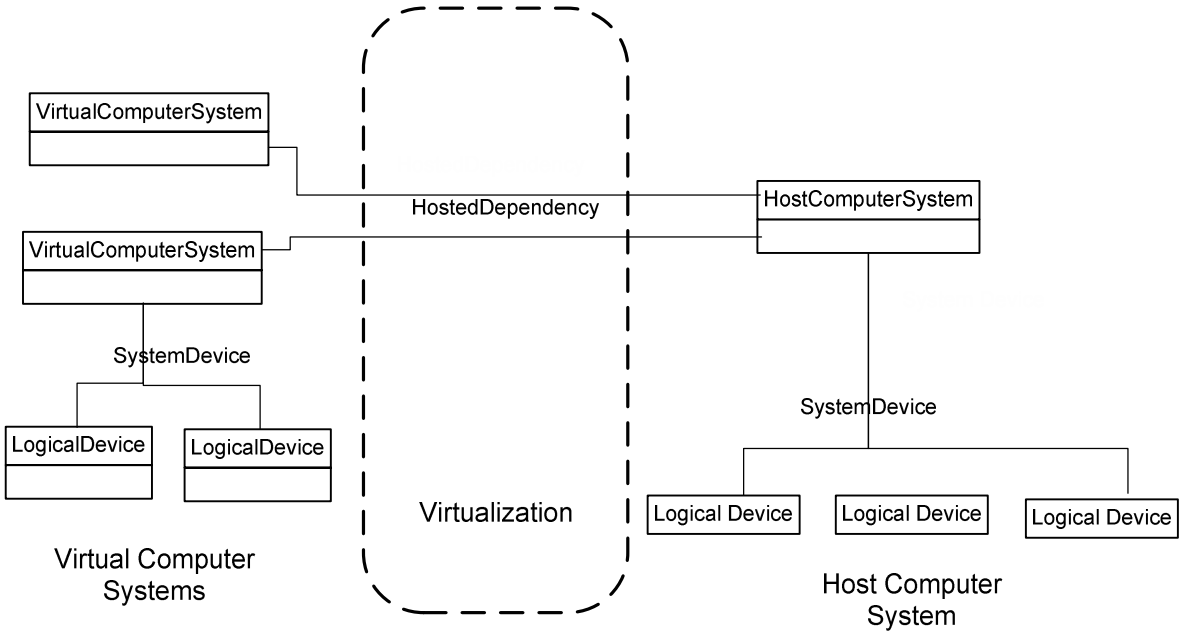


**Figure 2 Basic System Virtualization Model**

Additional instrumentation about the environment may be available outside the scope of the system virtualization model. The host computer system may provide management capabilities through the implementation of additional physical device and computer system profiles as defined in the Server Management Work Group (SMWG) or the Desktop & Mobile Work Group (DMWG). Additionally, the operating system or applications running in a virtual computer system may also implement aspects of CIM models. The information presented through this means (often described as “in band” or “through OS”) reflects the basic view of the resources for a single virtual system. Identifying correlating properties so that management clients can combine various instrumentation sources into a single unified view is an important requirement of the system virtualization and related modeling work. Figure 3 shows how the OS, virtualization and hardware models might be presented for this environment.

Control of the basic lifecycle operations (activate, deactivate, suspend) of the virtual system is available consistently for computer systems through the RequestStateChange method. For more details of the management of the virtual computer system lifecycle, see 3.2.3.1.

Subsequent sections describe the modeling details for resource virtualization, resource allocation, and virtual system configuration representation.

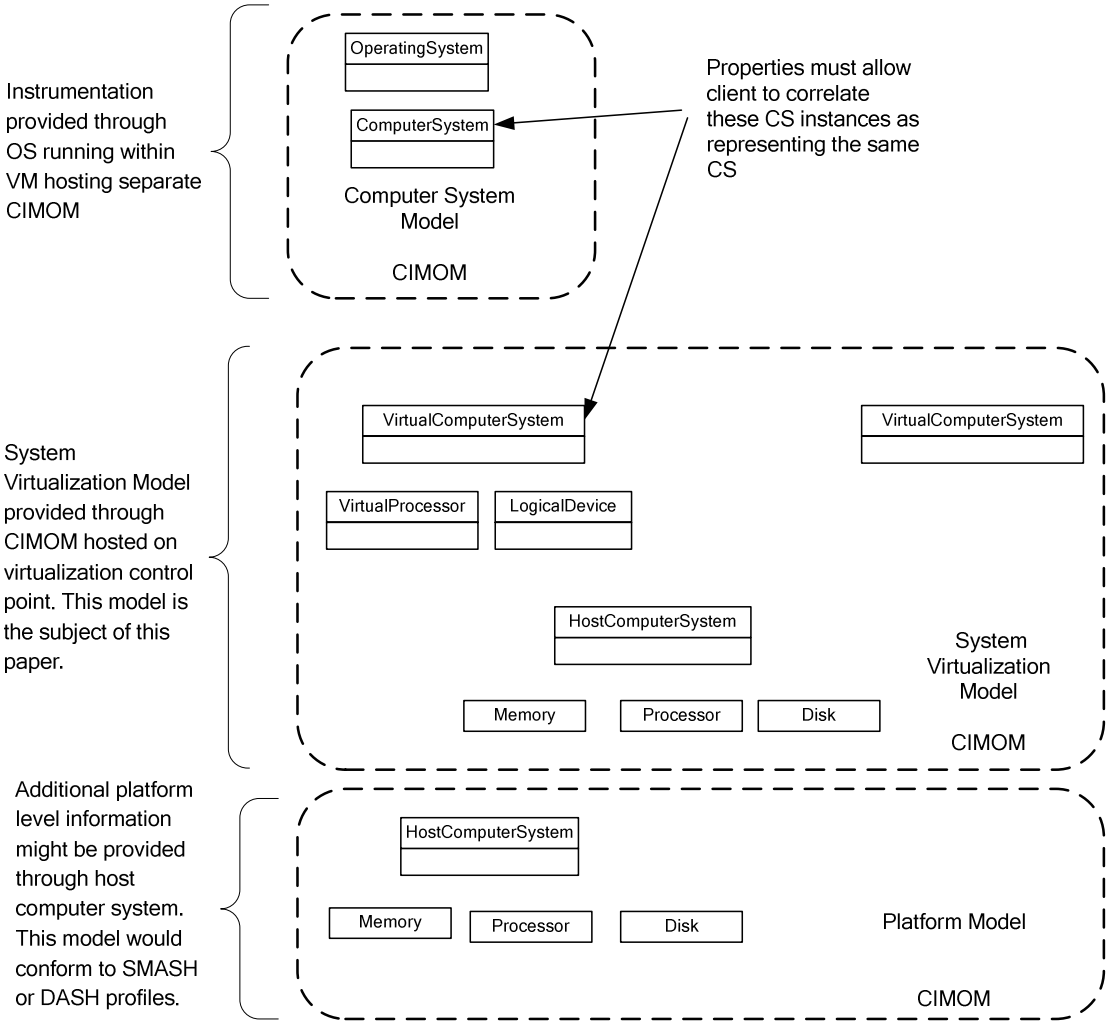


**Figure 3 Multiple CIMOMs View**

## 2.3 Modeling Virtual Devices and Systems

Figure 2 shows that a virtual computer system can be modeled as an instance of the CIM_ComputerSystem class with its devices modeled as instances of subclasses of CIM_LogicalDevice (CIM_Memory, CIM_ Processor, and so on). This model enables a management client to manage a virtual system without understanding the details of virtualization. To fully manage a virtual system environment the management client must have available additional information about the virtual computer system and related virtual devices. This additional virtualization-specific information is made available through an instance of a subclass of the CIM_SettingData class associated with the base device instance as a "state" extension or aspect. This basic pattern is shown in Figure 4. Instances of the CIM_LogicalDevice class representing virtual system logical devices are associated through the CIM_SettingsDefinesState association to related instances of CIM_ResourceAllocationSettingData class (RASD), which provides additional virtualization related details about the device. For example, information about the backing host device, the quantity of host resource allocated, and so on would be presented in the associated RASD

instance. Likewise, the CIM_ComputerSystem instance that represents the virtual computer system has an associated instance of the CIM_VirtualSystemSettingData class that provides additional information about the virtual computer system.

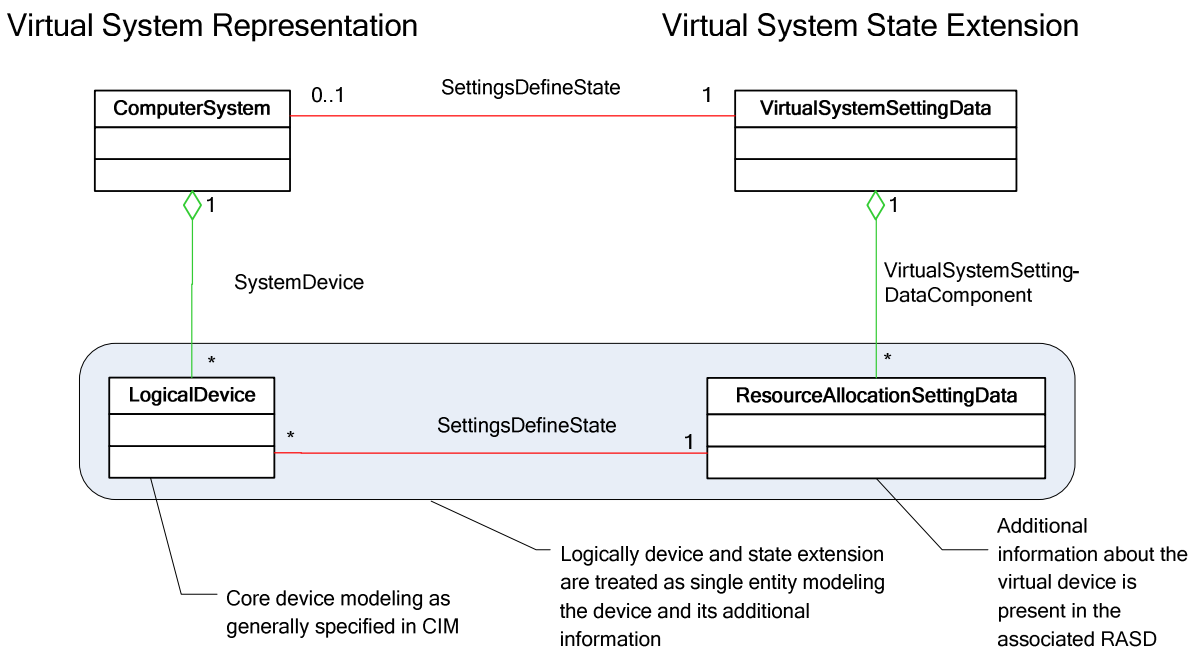


**Figure 4 Virtual System with Device and State Extension**

## 2.4 Virtual System Configurations

There are several contexts for which it is important to model a virtual system configuration, even if the virtual system is not currently active. Figure 5 illustrates the concept of a virtual system configuration that consists of setting data that represents the virtual system and an associated setting data instance for each of the configured resources. An instance of CIM_VirtualSystemSettingData represents the virtual computer system configuration information and instances of CIM_ResourceAllocationSettingData instances represent configuration information for each of the virtual devices.

A virtual system configuration is used to represent a saved virtual system configuration (for example, the configuration information that might be represented in a configuration file for an inactive virtual system).

Many system virtualization implementations support the functionality of snapshots. Snapshots capture the state of a virtual system, allowing the user to revert back to a snapshot that restores the complete state of the system to the state when the snapshot was captured. Each instance of a snapshot is modeled with a virtual system configuration that represents the state when the snapshot was taken.

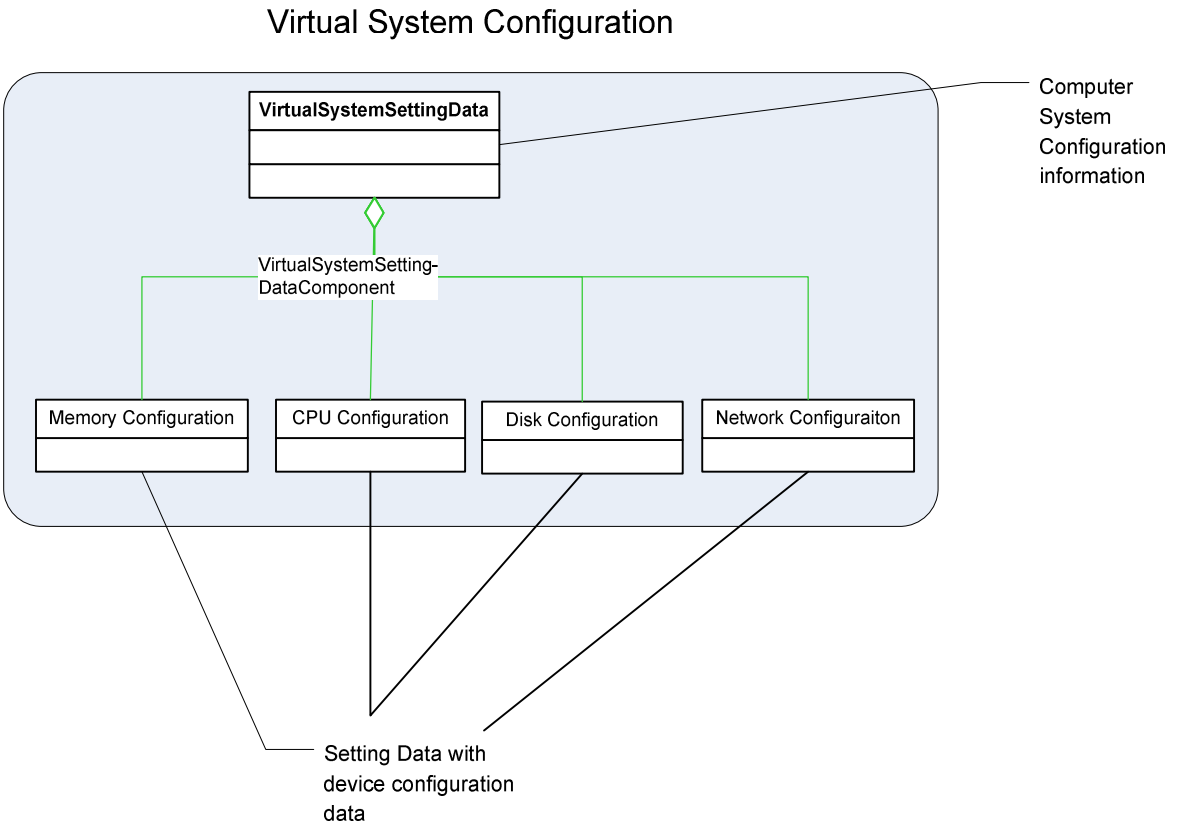


**Figure 5 Virtual System Configuration**

## 2.5 Modeling Resource Pools and Resource Allocations

Resource pools and resource allocations are the key elements for virtualization modeling. These elements are shown working together in Figure 6.

A resource pool is a logical entity (with associated controls) provided by the host system for the purpose of allocation and assignment of resources. A given resource pool may be used to allocate resources of a specific type. Pools may have associated host resources, but pools are not required to have component host resources. Resources allocated from a resource pool for virtual devices are represented by instances of the appropriate subclass of CIM_LogicalDevice with an associated instance of CIM_ResourceAllocationSettingData representing the allocation information.

The resource virtualization model provides for representing the relationship between a virtual device and its underlying host device through the CIM_LogicalIdentity or the CIM_HostedDependency association as long as that relationship is relatively static (like for example for disk devices). Often these relationships change very dynamically and it does not make sense to return this level of information (for example processor or memory resources).

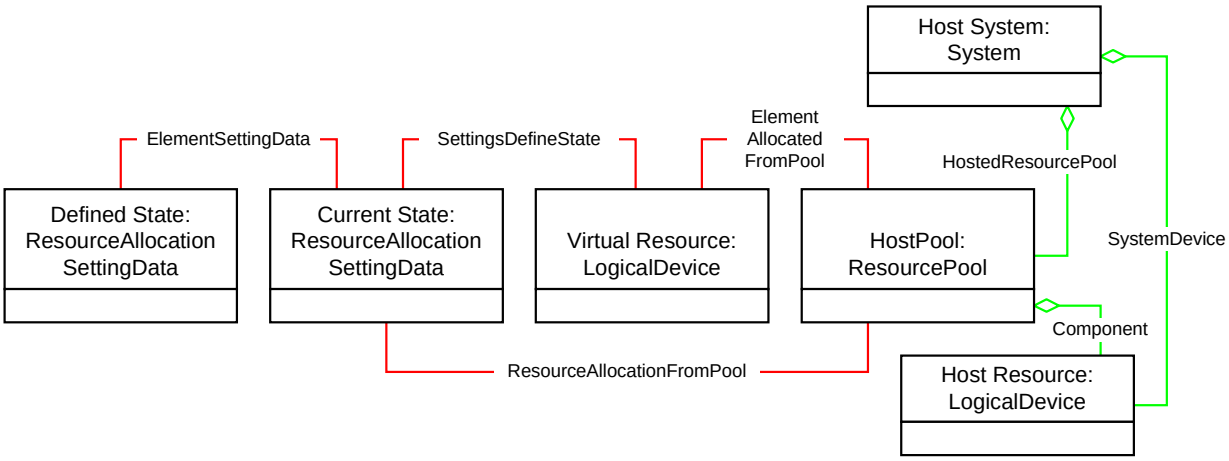


**Figure 6 Resource Pools and Resource Allocation**

# 3 The System Virtualization Model

This section provides details about profiles related to virtualization and the concepts behind the system virtualization model.

## 3.1 Profile Structure

In the DMTF, the CIM schema and associated behavior for a particular management domain is defined through a series of management profile documents. Each profile identifies the classes, properties, methods, and values that should be instantiated and manipulated to represent and manage a given domain. Figure 7 shows the structure of the profile documents related to virtualization. Two abstract profiles, *Resource Allocation Profile* and *Allocation Capabilities Profile*, describe the basic abstract patterns used for management of virtual systems. Two top-level, autonomous profiles specialize the *Computer System Profile*: *System Virtualization Profile* and *Virtual System Profile*. A series of device-specific profiles describe in more detail the management of virtual devices.

The *Resource Allocation Profile* describes the basic resource allocation pattern for resource pools, allocations, and setting data. It also defines the resource-pool-lifecycle management and relationships.

The *Allocation Capabilities Profile* extends the management capability of referencing profiles by adding the ability to represent the default, supported, and range of property values for resource allocation requests for a given resource, and the mutability of properties in a CIM_ResourceAllocationSettingData instance.

The *System Virtualization Profile* is an autonomous profile that specifies the 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.

The *System Virtualization Profile* references a set of component profiles that specify resource allocation for specific device types like CPU, memory, storage (block and file backed) and storage adapters, networking and networking adapters, removable devices, keyboard, video and mouse devices. These component profiles are specializations of both the *Resource Allocation Profile* and the *Allocation Capabilities Profile*.

The *Virtual System Profile* is an autonomous DMTF management profile that defines the model needed to provide for the inspection of a virtual system and its components. The *Virtual System Profile* specializes the *Computer System Profile* that defines the model needed to define a basic computing platform. In addition, the *Virtual System Profile* defines optional basic control operations for activating, deactivating, pausing, or suspending a virtual system.

The Computer System Profile references a set of component profiles that are defined for each of the device types that make up a computer system including CPU, memory, storage (block and file backed) and storage adapters, networking and networking adapters, removable devices, keyboard, video and mouse devices.

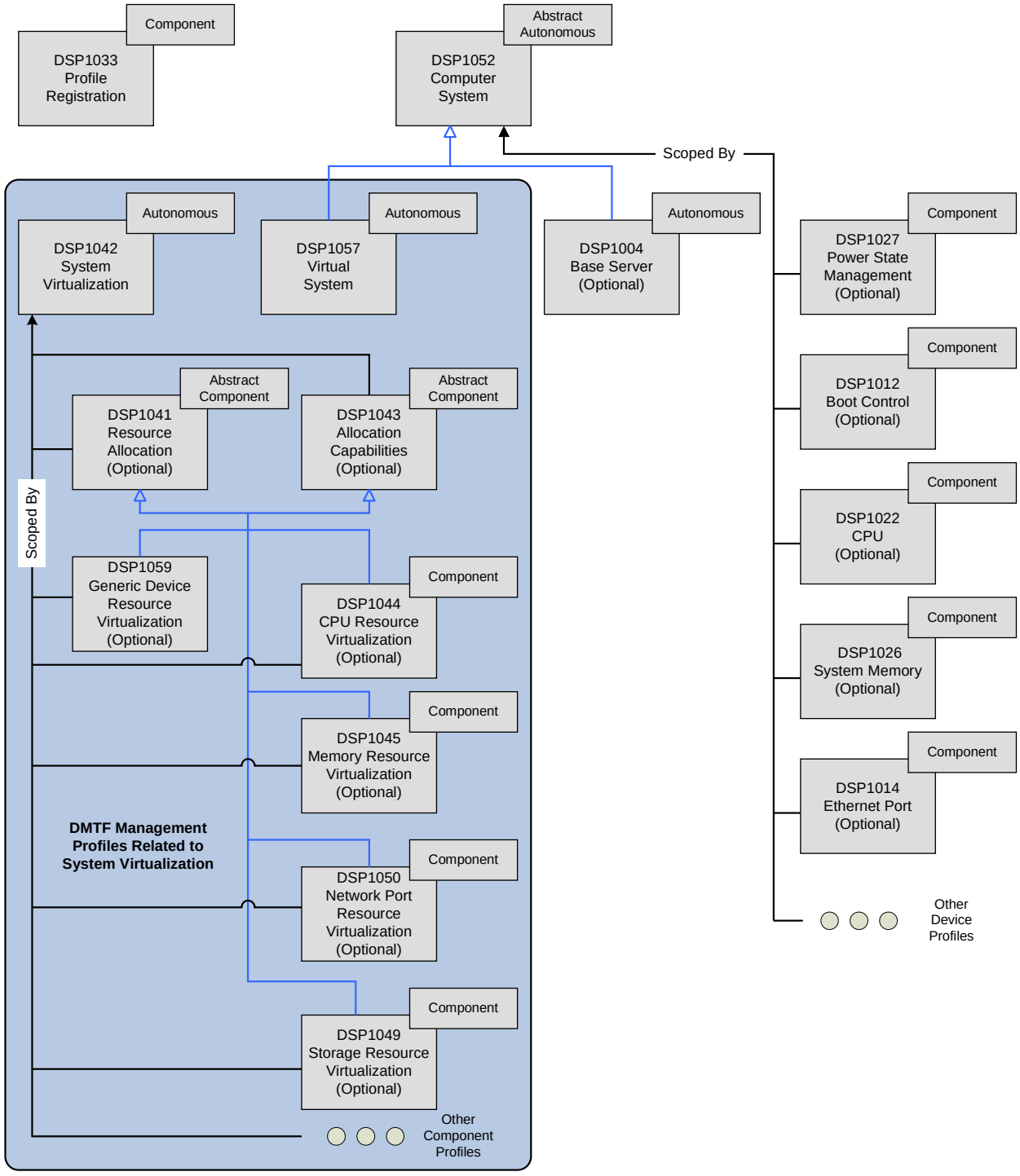


**Figure 7 Virtualization Profile Structure**

## 3.2 Concepts Addressed by the Model

Conceptually, the system virtualization model can be divided into the following components:

- Resource allocation that includes models for resource pools, resource allocation from pools, and services for managing pools.
- Allocation capabilities that provides the ability for a client to determine at runtime the system capabilities, including minimum, maximum and default values for resource allocation related properties.
- System virtualization and virtual systems that enables a client to manage virtual systems, including enumerating virtual systems and their component resources and controlling the lifecycle of virtual systems, and to manage the host computer system including creation of virtual systems and management of virtual system configurations including snapshots.
- Virtual device that extends existing device models by exploiting the resource allocation and allocation capabilities patterns to enable management of virtual devices.

### 3.2.1 Resource Allocation

The classes for the resource allocation model are shown in Figure 8. The main classes are CIM_ResourcePool class and CIM_ResourceAllocationSettingData class as well as the classes modeling the capabilities and service classes for the manipulation of pools.

A resource pool, modeled using the CIM_ResourcePool class, is the central management point for the allocation of resources. Typically a pool collects host system resources whose capabilities are allocated to a consumer. Resources allocated from pools with no component devices are known as “synthetic devices.” In many implementations, virtual Ethernet adapters are an example of a synthetic device.

Types of allocation supported include

- pass-through resource allocation, in which the allocated device is identical to the pool device
- dedicated resource allocation, in which the virtual device is allocated exclusive use of the pool device
- shared resource allocation, in which the allocated device is shared among consumers

Properties of the CIM_ResourcePool class model information about the resource type and capacity supported by the pool.

Allocation from a pool is represented by an instance of CIM_ResourceAllocationSettingData. As mentioned previously, this key class in the virtualization model represents allocations, as well as extended state and capabilities. Properties of this class model provide information allowing the client to determine the type and quantity of resource consumed, as well as the quantity of virtual resource exposed and the type of allocation.

Initially host computer system resources are aggregated into a “primordial” pool indicated by a property in the CIM_ResourcePool class.

Some implementations may support creating and managing child pools and moving resources in and out of pools. These functions are modeled using the CIM_ResourcePoolCapabilities and CIM_ResourcePoolService classes.

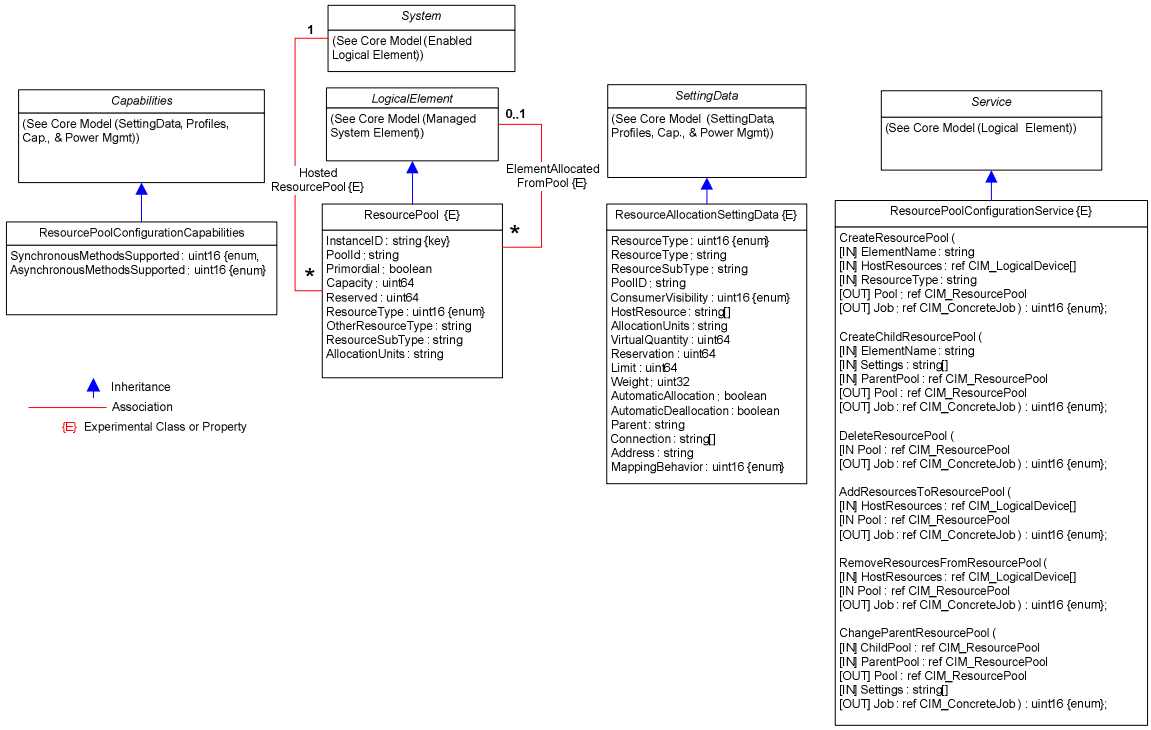


**Figure 8 Resource Allocation Class Diagram**

In Figure 9, we see an example of these resource management classes showing an instance of a resource pool that collects host system devices. Allocated from the pool are virtual devices and RASDs, which give additional information about the allocation.

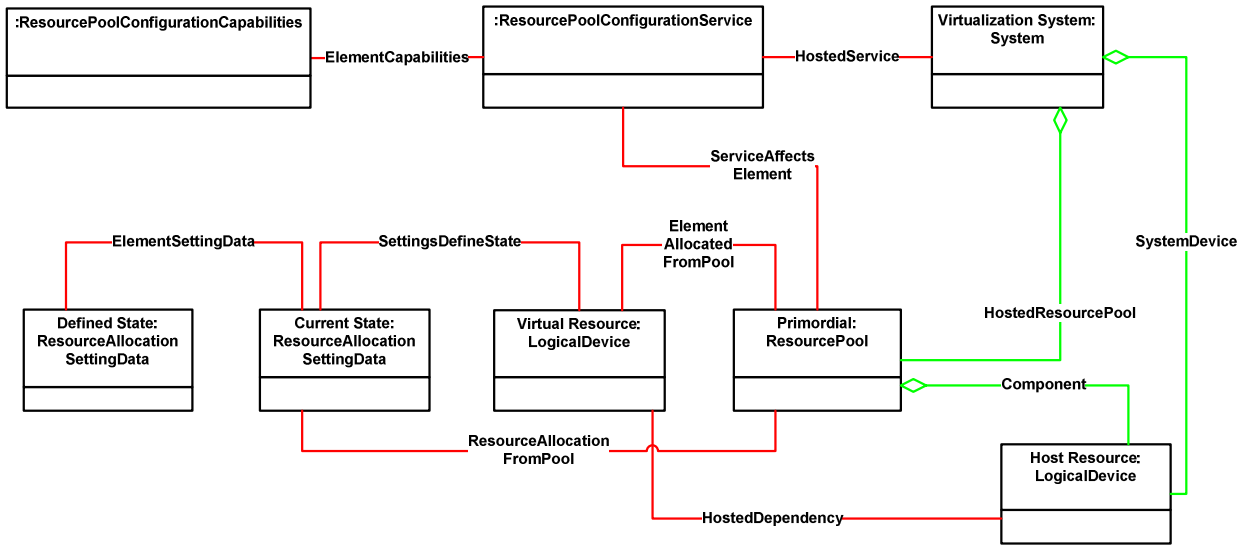


**Figure 9 Virtual Resource Allocation Instance Diagram**

In Figure 10 another use of these classes is shown in an example of "simple" resource allocation. In this case there is no allocated logical device; the allocated resources are shown only as RASDs. This pattern is appropriate where the allocated resource is not modeled as a device, as in electrical power allocation, for example.

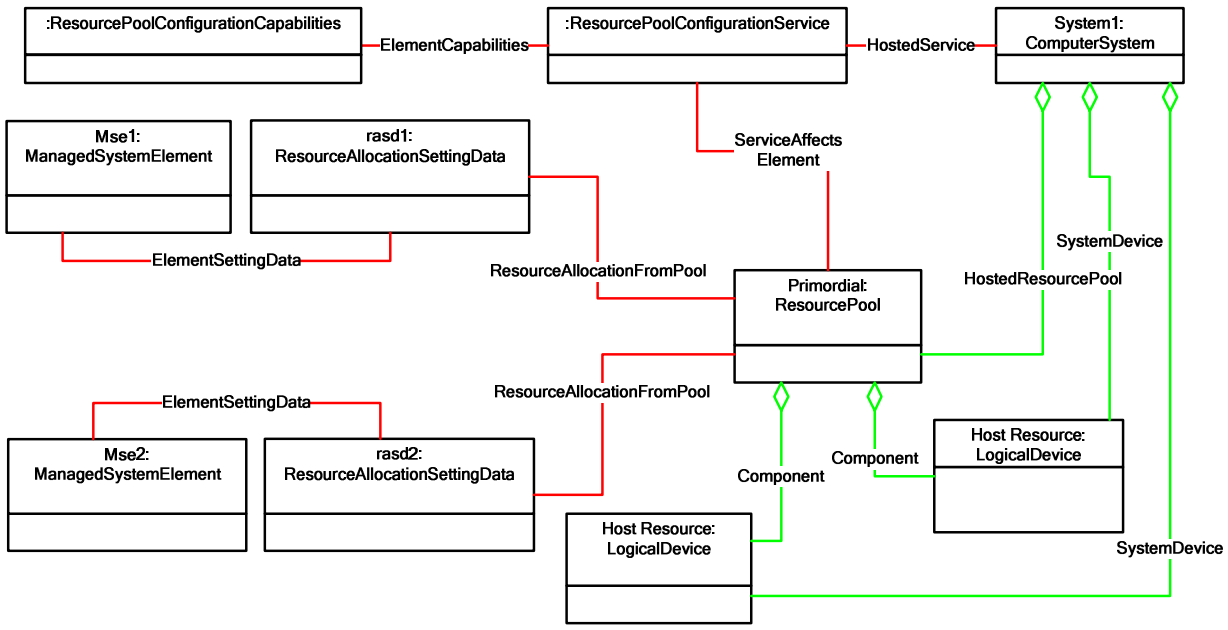


**Figure 10 Simple Resource Allocation**

### 3.2.2 Allocation Capabilities

The basic classes associated with the allocation capabilities model are shown in Figure 11. This pattern enables a client to determine at run time the capabilities, including minimum, maximum, default, and specific values that are supported by the implementation in various contexts.

The basic pattern uses an instance of the CIM_AllocationCapabilities class and a collection of RASDs associated through the CIM_SettingsDefinesCapabilities association whose properties values are set to define the role (minimum, maximum) of the values in the associated RASD. This idea is best illustrated in the following examples.

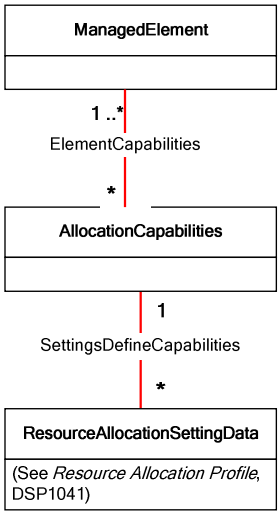


**Figure 11 Allocation Capabilities Class Diagram**

Figure 12 shows the allocation capabilities pattern applied at the host computer system and at the resource pool. A capability set (an instance of CIM_AllocationCapabilities and the associated instances of CIM_ResourceAllocationSettingData) at the host computer system level applies to all resources of the specified type. A capability set associated with a resource pool would apply to resources created from the resource pool.

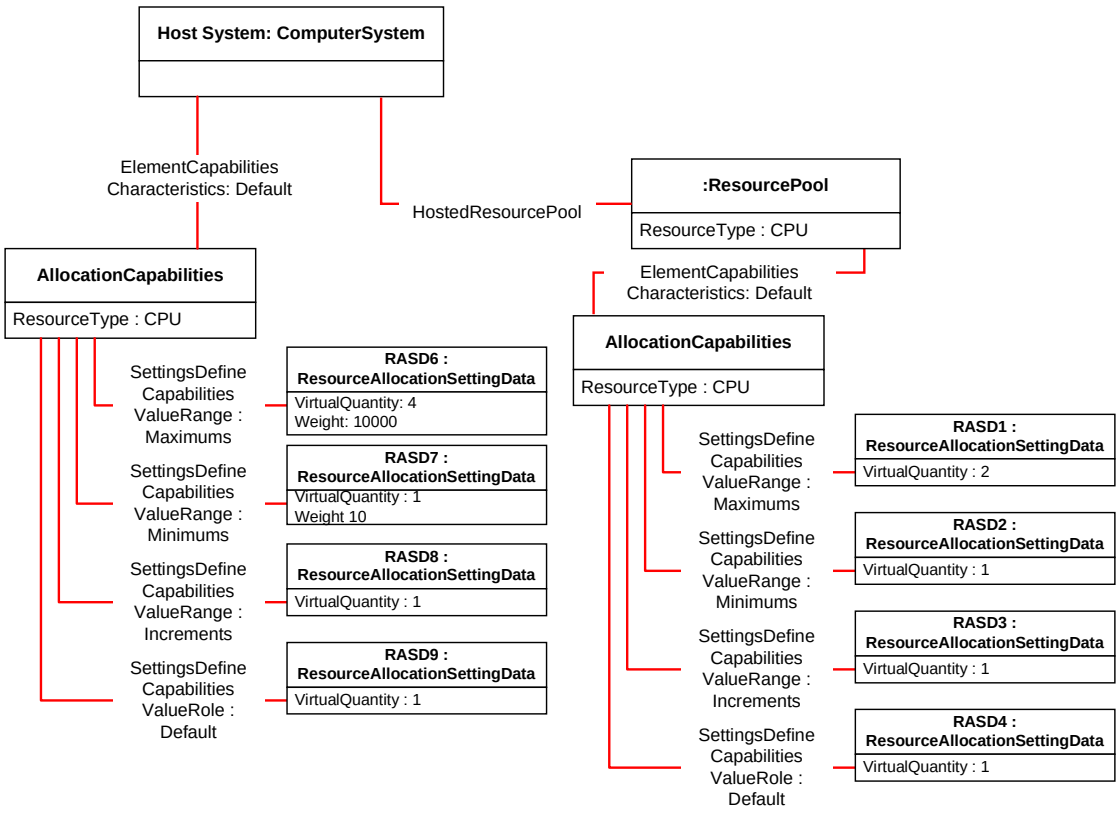


**Figure 12 Allocation Capabilities Applied to Host Computer System and Resource Pool**

The pattern can also be used to understand mutability of a virtual resource as shown in Figure 13. In this case the capability set specifies the ranges for changing the virtual quantity and weight of a virtual system resource allocation.

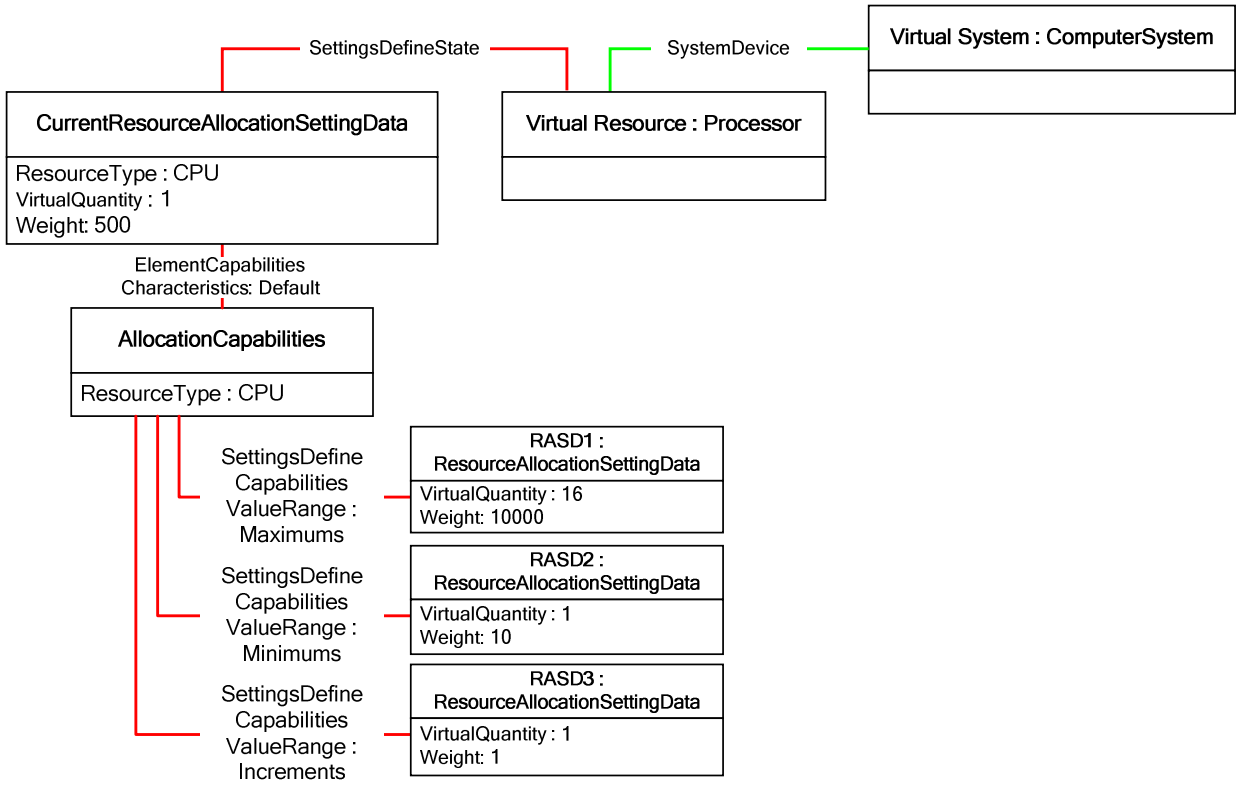


**Figure 13 Allocation Capabilities Applied to a Virtual System Resource Allocation**

### 3.2.3 System Virtualization and Virtual Systems

The classes introduced in the modeling of virtual systems are shown in Figure 14. The CIM_VirtualSystemManagementService and CIM_VirtualSystemManagementCapabilities classes provide the ability to add, delete, and modify resources of a virtual system and to define and delete a virtual system.

Clients can determine specific information about an implementation's support for virtual system resource manipulation through the allocation capabilities instances that are associated with the virtual resources and resource pools.

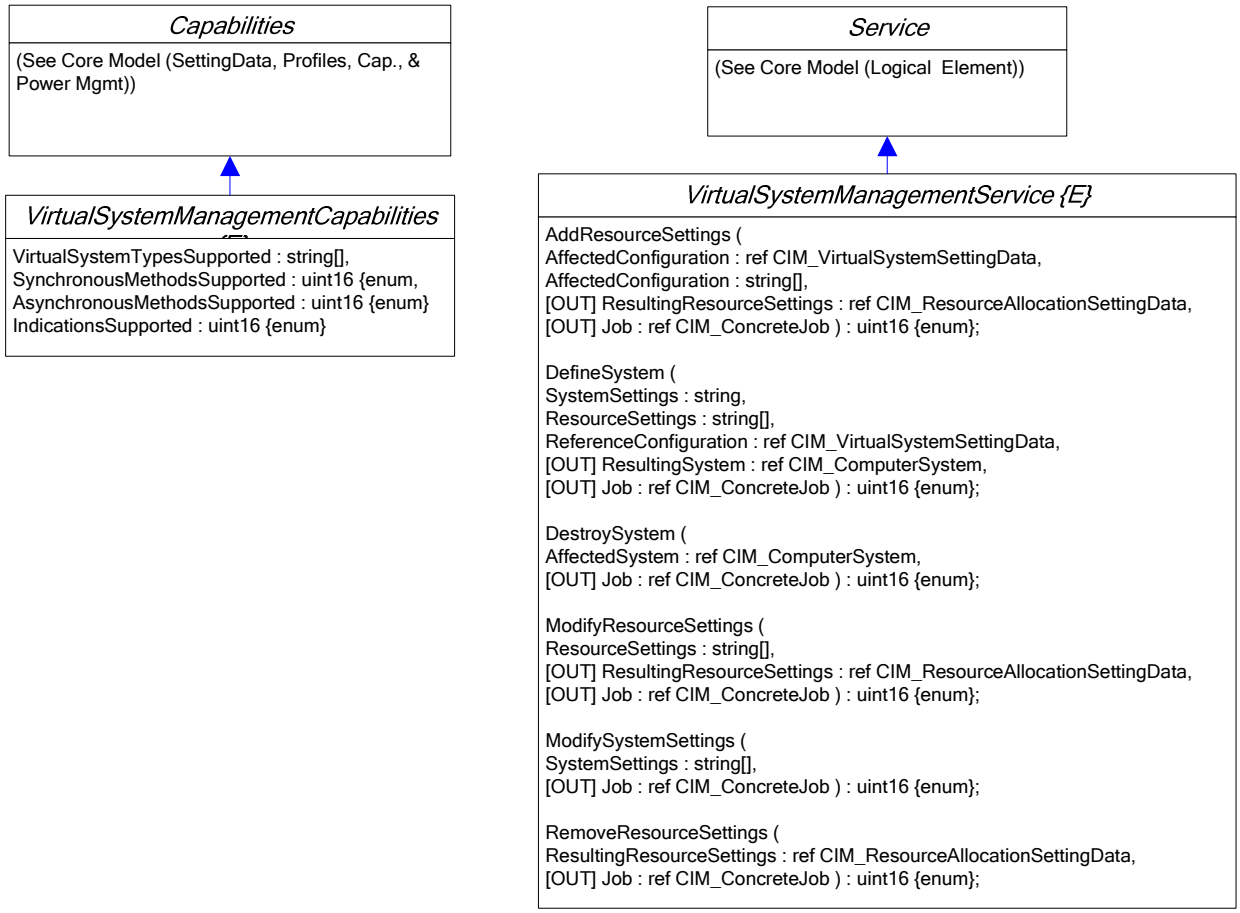


**Figure 14 Virtual System Modeling Class Diagram**

### 3.2.3.1 Virtual System States

The states of a virtual system are shown in Figure 15.

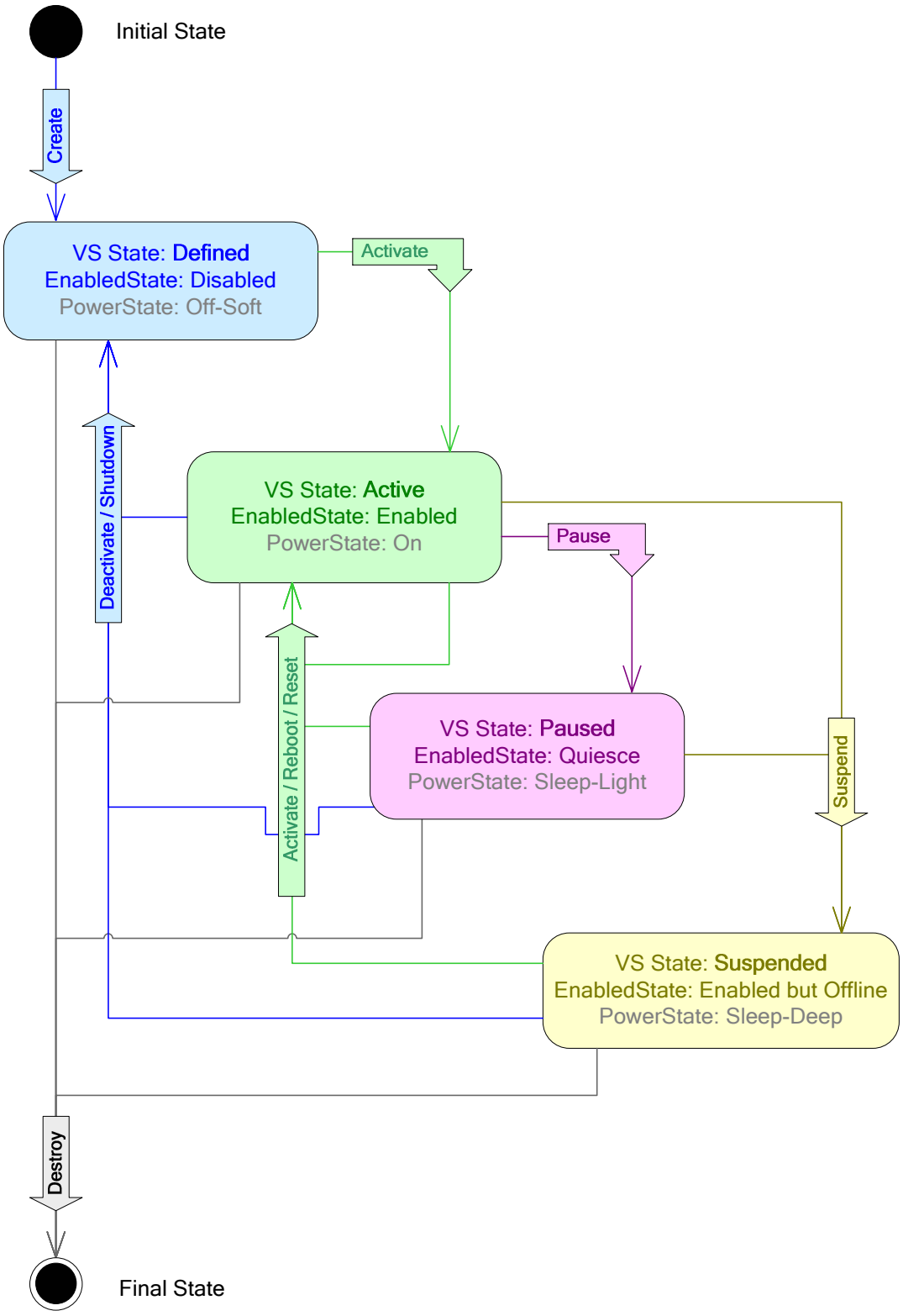


**Figure 15 Virtual System State Diagram**

A new virtual system can be created by using the DefineSystem( ) method of the CIM_Virtual System ManagementService. Inputs to this method are CIM_VirtualSystemSettingData and CIM_ResourceAllocationSettingData instances, which define the virtual system as a whole and each of the desired virtual resources. At the successful completion of this method the new virtual computer system is represented in the model by an instance of the CIM_ComputerSystem class (with state extension) and a virtual system configuration that represents the saved configuration information. Any devices that are allocated during definition (for example, virtual disk) are represented by the appropriate logical device. The system is in the "Defined" state. This would also correspond to a "Powered Off" state. A diagram of an example system in this state is shown in Figure 16.

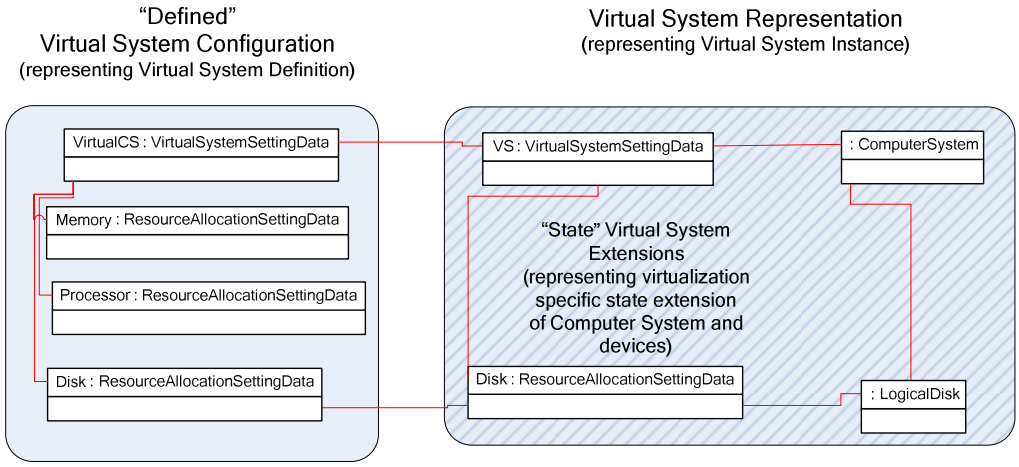


**Figure 16 Defined Virtual System Representation**

From the "Defined" state a system can be activated to enter the "Active" or "Powered On" state. During activation the underlying system allocates resources as specified in the virtual system configuration setting data instances, and device and state extension CIM instances are instantiated. The resulting virtual system is modeled as shown in Figure 17.

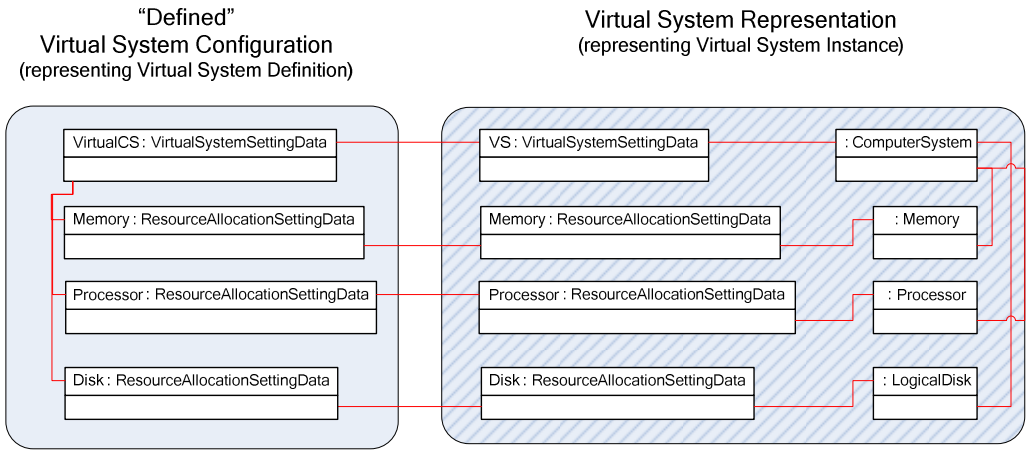


**Figure 17 Active Virtual System Representation**

### 3.2.4 Virtual Device Modeling

The classes and pattern used for virtual device modeling have already been introduced. An instance of a virtual device is represented by the appropriate CIM_LogicalDevice subclass, the allocation and state are represented using RASDs, and the capabilities for allocation and modification of the device are represented using the Allocation Capabilities pattern previously described. An example instance diagram illustrating these concepts is shown in Figure 18.

Typically there will be a device profile that describes the behavior of the device model in general (for example, *CPU Profile*, *System Memory Profile*, and so on) and a virtualization-related profile that describes additional considerations for modeling virtual devices using the virtualization patterns (*Processor Device Resource Virtualization Profile*, *Memory Resource Virtualization Profile*, and so on).

The *Generic Device Resource Virtualization Profile* provides a general profile that can be applied if more specific device or device virtualization profiles are not available.

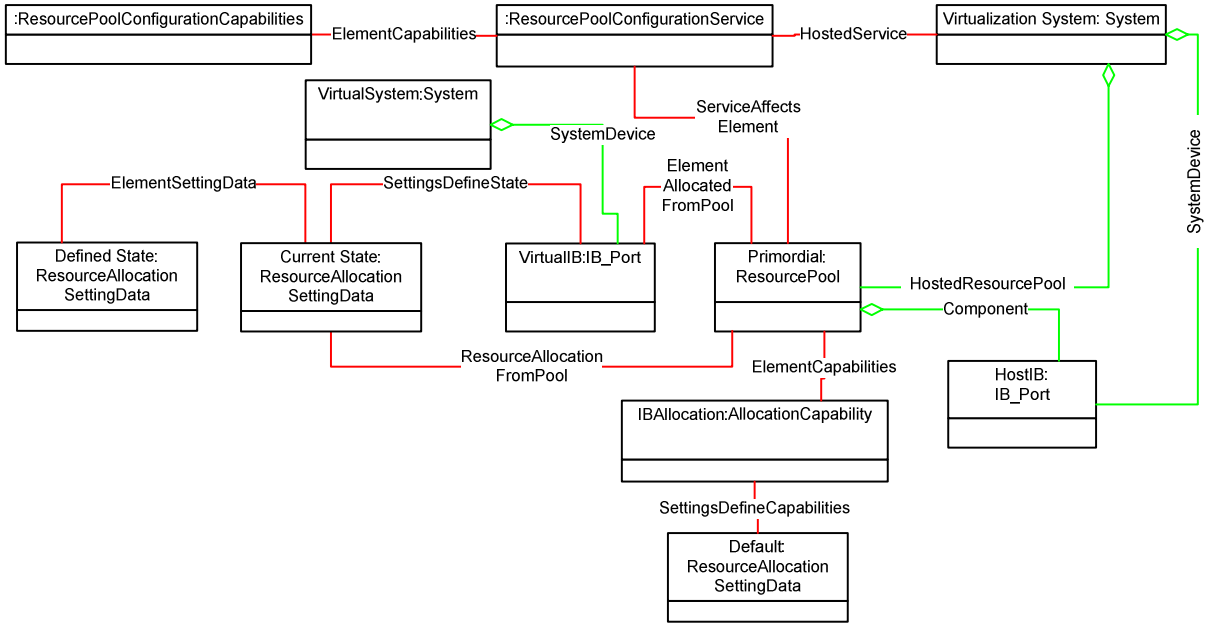


**Figure 18 Instance Diagram for Virtual Device Model**

# 4 Relationships to Other Standards and Specifications

## 4.1 Overlapping Standards and Specifications

There are no known virtualization management standards. This work extends the existing CIM system modeling by reusing system and logical device classes to model virtual systems.

# 5 System Virtualization Model Use Cases

This section provides use cases for managing the host computer system or a virtual computer system.

## 5.1 Managing the Host Computer System

The System Virtualization model is defined to allow a management client to determine at execution time information about the managed virtualization environment, including supported virtual resource types, valid values for resource allocations, and capabilities for managing resource pools.

### 5.1.1 Discovering the CIM implementation for a System Virtualization

A client can discover CIM implementations of virtualization management through SLP, by following profile registration associations, or through *a priori* knowledge of host name or IP address where CIMOM is running.

### 5.1.2 Discovering a Host Computer System

A client can find instances of the CIM_ComputerSystem class representing host systems can be found by following the CIM_ElementConformsToProfile association from the instance of the CIM_RegisteredProfile class representing *System Virtualization Profile*

### 5.1.3 Determining the Capabilities of an Implementation

To determine the capabilities of an implementation, from the instance of CIM_ComputerSystem that represents the host computer system the client can traverse the CIM_ElementCapabilities association to an instance of CIM_VirtualSystemManagementCapabilities. Properties of this instance supply information about supported virtual system types, methods, and indications.

### 5.1.4 Determining the Supported Resource Types of an Implementation

The preferred mechanism for determining supported resource types is to find instances of registered profiles scoped by the *System Virtualization Profile*, find the central class for each of these profiles, and collect the resource types represented..

### 5.1.5 Finding Resource Pools and their Constituent Resources

A client can find Resource Pools by traversing the the CIM_HostedResourcePool association from the instance of CIM_ComputerSystem that represents the host computer system. A client can identify elements of a resource pool by traversing the CIM_Component association to the appropriate subclass of the CIM_LogicalDevice class.

### 5.1.6 Determining the Capacity and Allocation of a Resource Pool

The Capacity property of the CIM_ResourcePool instance that represents the resource pool provides the total capacity of this pool. The Reserved property provides the total amount of the currently allocated resources.

### 5.1.7 Determining Resources Allocated from a Resource Pool

To determine the details of the resources allocated from a given resource pool a client can traverse the CIM_ElementAllocatedFromPool association to find all of the devices allocated from this pool. A client can determine additional details about the allocation by traversing the CIM_ResourceAllocatedFromPool association to each RASD for the allocations.

### 5.1.8 Determining the Valid Settings for a Resource Allocation

An instance of CIM_AllocationCapabilities and its associated (through CIM_SettingsDefinesCapabilities) CIM_ResourceAllocationSettingData instances can help a client determine the capabilities for a resource type or a resource type allocated from a specific resource pool. The client can find the CIM_AllocationCapabilities instance by following CIM_ElementCapabilities from an instance of CIM_ResourcePool or CIM_ComputerSystem.

### 5.1.9 Locating Virtual Systems Hosted by a Host Computer System

Given a CIM_ComputerSystem instance that represents the host computer system, the CIM_ComputerSystem instances associated to the host computer system through CIM_HostedDependency are the instances of virtual computer systems.

## 5.2 Managing a Virtual Computer System

The following use cases show various aspects of managing a virtual computer system.

### 5.2.1 Creating a Virtual Computer System

The basic operation of creating a virtual computer system is done using the DefineSystem( ) method of the CIM_VirtualizationManagementService associated with the instance of CIM_ComputerSystem that represents the host computer system. This method takes as input an instance of CIM_VirtualSystemSettingData and an array of instances of CIM_ResourceAllocationSettingData, which represent the requests for resources that are required to compose the target virtual computer system. Valid resource types and ranges of property values can be determined as noted in previous use cases. Typically, implementations also provide default values for most property values if they are unspecified.

### 5.2.2 Determining a Virtual System's State and Other Properties

From the instance of CIM_ComputerSystem that represents the virtual computer system the EnabledState property represents the virtual system's state. Other properties of the virtual system can be obtained from the CIM_ComputerSystem instance and from the associated (through the SettingsDefineState association) CIM_VirtualSystemSettingData (VSSD) instance.

### 5.2.3 Determining the "Defined" Virtual System Configuration

Each virtual computer system has a "Defined" configuration that is permanently recorded and takes effect when a deactivated virtual computer system is activated. This configuration can be determined given an instance of CIM_ComputerSystem that represents the virtual computer system by following the SettingsDefineState association to the VSSD instance, which represents the state extension for the virtual computer system. From this VSSD instance the

CIM_ElementSettingData associated VSSD anchors the “Defined” virtual system configuration, and each of the associated RASDs provides details of the virtual system device configuration.

### 5.2.4 Determining the Virtual System Structure

From the instance of CIM_ComputerSystem that represents the virtual computer system the client can traverse the CIM_SystemDevice association to find the component devices of the virtual computer system. For each of these devices the associated RASDs that represent the virtualization “extensions” of the device can be found by the CIM_SettingsDefineState association, and persistent device configuration information can be obtained by RASD associated to the state extension by the CIM_ElementSettingData association.

### 5.2.5 Changing the Virtual System State

Given a reference to the CIM_ComputerSystem instance that represents the virtual computer system the client can effect state changes by invoking the CIM_EnabledLogicalElementRequestStateChange( ) method. A client can determine valid state values by finding the associated CIM_ElementCapabilities instance and analyzing the RequestedStatesSupported property.

### 5.2.6 Modifying a Virtual System

Clients may be able to modify various aspects of a virtual computer system including adding or deleting virtual resources, modifying virtual resource definitions, or modifying the virtual resource state extension. Allocation capabilities and associated RASDs can be used to determine what modifications an implementation supports.

To add new virtual resources the client prepares one or more instances of RASDs that represent the allocation requests for new virtual resources and invokes the AddResourceSettings( ) method on the virtual system management service to add the virtual resources.

To modify existing resources the client obtains RASDs or VSSDs that represent the state extension or resource definition to be modified and alters the properties in these instances locally within ranges supported by the implementation. The ModifyResourceSettings( ) method of the virtual system management service can then be invoked with the modified RASD values to effect the desired changes.

### 5.2.7 Destroying a Virtual System

Given a reference to an instance of CIM_ComputerSystem that represents the virtual system to be destroyed, the DestroySystem( ) method on the virtual system management service can be invoked to effect the virtual system destruction.

### 5.2.8 Managing Snapshots

The following use cases illustrate various aspects of snapshot management.

#### 5.2.8.1 Determining Support for Snapshots

Properties of the CIM_VirtualSystemSnapshotServiceCapabilities that are associated with the CIM_VirtualSystemSnapshotService can be examined to determine the level of support for snapshot types and snapshot related methods.

#### 5.2.8.2 Creating a Snapshot

The CreateSnapshot( ) method on the VirtualSystemSnapshotService is invoked, passing a reference to the system that is the target for the snapshot.

#### 5.2.8.3 Locating Snapshots of a Virtual System

Given an instance of CIM_ComputerSystem that represents a virtual computer system the client can follow instances of the CIM_SnapshotofVirtualSystem association to instances of CIM_VirtualSystemSettingData, which anchors the snapshot related configuration classes for a snapshot.

#### 5.2.8.4 Locating the Most Current Snapshot in a Branch of Snapshots

Given an instance of CIM_ComputerSystem that represents a virtual computer system, the client can follow the CIM_MostCurrentSnapshotInBranch association to the instance of CIM_VirtualSystemSettingData that represents the most recent snapshot.

#### 5.2.8.5 Locating Dependent Snapshots

Given a reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot, the client can follow the CIM_Dependency association to instances of CIM_VirtualSystemSettingData that represent dependent snapshots (if any).

#### 5.2.8.6 Applying a Snapshot

Given a reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot, the client invokes the ApplySnapshot( ) method on the virtual system snapshot service. The system is deactivated and system state is restored to the value of the resources represented in the snapshot. The system is then activated.

#### 5.2.8.7 Destroying a Snapshot

The DestroySnapshot( ) method on the VirtualSystemSnapshotService is invoked, passing a reference to the CIM_VirtualSystemSettingData instance that represents the snapshot to be deleted.

# Appendix A – References

[1] Common Information Model (CIM) Specification, 2.2, June 14, 1999 - Downloadable from http://www.dmtf.org/spec/cims.html

[2] Unified Modeling Language (UML) from the Open Management Group (OMG) - Downloadable from http://www.omg.org/uml/

# Appendix B – Extending the Model

The system virtualization model was designed for extensibility in several areas, including the following:

- Virtual resource types can be added by adding a resource type value from the Vendor Reserved range of the ResourceType property of the CIM_ResourceAllocationSettingData, CIM_ResourcePool, and CIM_ResourceAllocation classes. The Description property is used to provide additional details of the new resource type.
- Virtual resource subtypes can be added using the ResourceSubtype property for the associated classes. For example, the ResourceSubtype property can be used to distinguish different models of a particular resource.
- The CIM_ResourceAllocationSettingData class can be extended with implementation-specific properties. One of the main motivators for the use of instances of this class in several contexts is to allow the RASD to be extended once and then leveraged across the model. A RASD with vendor extensions is used in resource allocation, virtual system configuration and extended state and as part of the allocation capabilities model. By instantiating appropriate instances of RASD that are associated with instances of CIM_AllocationCapabilities the implementation can inform the management client about minimum, maximum and default values for a property that the client was not originally aware of when designed.