Document Number: DSP0215

Date: 2009-04-23

Version: 1.0.0

Server Management Managed Element Addressing Specification

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

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

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patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a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

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

CONTENTS

Figures

Tables

83

Foreword

The *Server Management Managed Element Addressing Specification* (DSP0215) was prepared by the Server Management Working Group of the DMTF.

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

Acknowledgments

The authors wish to acknowledge the following contributors and participants from the DMTF Server Management Working Group:

- Aaron Merkin – IBM
- Jeff Hilland – HP
- Jon Hass – Dell
- Khachatur Papanyan – Dell
- Radhakrishna Dasari – Dell
- Jeff Hilland – HP
- Christina Shaw – HP
- John Leung – Intel

Introduction

This document describes the Server Management (SM) Managed Element (ME) addressing standard. SM ME addressing provides an easy, user friendly way to address Common Information Model (CIM) objects (classes and instances). This specification may be used to define valid targets for SM CLP commands.

Overview

CIM is a rich modeling environment in that it provides many ways to model computing environments. It allows different vendors to choose different model components to model their computing equipment and environments. While this open-ended modeling behavior is desirable in the theoretical sense, it poses difficulties in a practical environment in that it often creates a situation in which different vendors' tools cannot interoperate because they have different notions of the computing model. SM ME addressing provides an approach to avoid these difficulties.

SM ME addressing is based on the CIM containment hierarchy described in section 5. The containment hierarchy defines the hierarchical CIM container-based addressing associations that are combined to form a fully qualified address path to an instance, much like the path to a file in a file system. The containment hierarchy, which is based on relevant ME classes and associations, forms the basis of the addressing grammar described in section 6. The SM profile specifications define the specific CIM objects that may be addressed and their supported non-addressing associations. The *SM CLP Specification* defines how to use an SM ME address in a CLP command and how to discover what non-addressing associations are supported for any given target ME.

A fully qualified address path (called a UFiP) to a User Friendly instance Tag (UFiT) addresses a single CLP command target and may be mapped to a specific CIM object reference, thus providing support for communication between the Manageability Access Point (MAP) and CIM servers, clients, and providers. These features combine to enable an embedded lightweight CIM server in the Manageability Access Point.

Several key components form the technical foundation that enables SM ME addressing to provide a simple, user friendly, unique address path to a CIM instance. These key components are as follows:

- SM profiles
- SM ME grammar
- CIM model

The following sections discuss the role that each of these key technologies plays in the SM addressing scheme.

SM Profiles

The SM containment hierarchy defined in section 5 forms the basis for SM ME addressing by identifying the hierarchical containment associations between objects. The SM profiles define the model components that implementations must support. The SM ME addressing rules define how implementations create UFiTs for instances of classes specified in the SM profiles.

SM profiles define rules and guidelines to model specific hardware platforms such as base systems, modular (blade) systems, and other architectural features such as clustering, redundancy (resource sharing), and virtual machines. Other SM profiles define rules and guidelines to model specific management domains such as Boot Control, Power Control, Firmware Update, and the like. This addressing specification has the grammar and rules associated with the SM profiles.

If implementations follow the models specified in the profiles and the addressing rules, then fully qualified UFiPs are guaranteed to exist for all managed instances in the implementations' namespace. The SM profiles guarantee that all conforming implementations provide the same MEs and supporting associations. The SM containment hierarchy guarantees that these MEs are addressable.

SM ME Grammar

The SM ME addressing grammar described in section 6.3 provides unambiguous productions that guarantee that a programmatic parser can be coded to detect valid ME addresses. Furthermore, it guarantees that each valid production eventually terminates in a unique UFiT or User Friendly class Tag (UFcT). The difference between the UFcT and the UFiT is the unique instance suffix, which identifies a particular CIM instance of a class.

CIM Model

The CIM schema defines the valid associations between CIM instances. This specification provides the rules for subsets of CIM associations called addressing associations, which are used to form ME addresses. These rules are used to validate terms in a UFiP. The UFiTs on each side of the association path must have the proper object type and role. CIM defines the properties for each object class and its subclasses.

CIM was chosen as the underlying data model because of its capacity to normalize computer-based management relationships and information. The CIM schema describes a well-defined structured containment hierarchy that can be adapted to provide an unambiguous method of addressing MEs.

Target Audience

This specification is intended to guide developers of SM ME addressing implementations. It may also be used as a reference by system administrators and other users of an SM ME addressing implementation.

SM ME Addressing Goals

The goals of the *Server Management Managed Element Addressing Specification* are as follows:

- Provide a user friendly way to accurately address CIM objects, using a hierarchical containment structure based on specified CIM associations. This addressing mechanism is intended to be applicable to other DMTF protocols.
- Provide access to information in other Managed Element instances associated with the target ME instance through non-addressing associations that are not part of a SM containment hierarchy. This feature is required to support *n* dimensional association traversal rooted at the terminating target UFiT. *n* is bounded by the non-addressing associations defined in the SM profiles supported by the MAP. Addressing associations are defined in 5.4.
- Provide an unambiguous grammar to aid in programmatic parsing of a UFiP. The grammar is defined in 6.3.

Server Management Managed Element Addressing Specification

1 Scope

This document describes the Server Management (SM) Managed Element (ME) addressing standard. SM ME addressing provides an easy, user friendly way to address Common Information Model (CIM) objects (classes and instances). This specification may be used to define valid targets for SM CLP commands.

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, *Common Information Model (CIM) Schema*, 2.10, http://www.dmtf.org/standards/cim

DMTF DSP0214, *SM Command Line Protocol Specification*, 1.0.0, http://www.dmtf.org/standards/published_documents/DSP0214.pdf

IETF RFC 2234, *Augmented BNF for Syntax Specifications: ABNF*, November 1997, http://www.ietf.org/rfc/rfc2234.txt

OMG, *Unified Modeling Language (UML) from the Open Management Group (OMG)*, http://www.uml.org/

2.2 Other References

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

3 Terms and Definitions

For the purposes of this document, the following terms and definitions 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 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
shall
indicates requirements to be followed strictly in order to conform to the document and from which no deviation is permitted

3.9
shall not
indicates requirements to be followed in order to conform to the document and from which no deviation is permitted

3.10
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

3.11
should not
indicates that a certain possibility or course of action is deprecated but not prohibited

3.12
Addressing Association
a certain CIM association that is valid for the purpose of addressing CIM instances defined in SM profiles These CIM associations are based on the SM containment hierarchy. In the SM addressing grammar this association is denoted by a forward slash (/) or a backward slash (\) in the ME's UFiP. These associations are enumerated in Table 12 and Table 13, which define the logical and physical address containment.

3.13
Address Path
a path in which each term has the appropriate intervening addressing association
This is explained in detail in 5.4 and 6.2.

3.14
client
a logical component that manages a system through a Manageability Access Point (MAP)
A client may sometimes be referred to as the managing entity or management client.

3.15

instance suffix

a positive non-zero integer appended to the end of a UFcT to create an instance tag or UFiT

The instance suffix creates a unique identifier for the instance within the defining container. A UFcT plus an instance suffix is a UFiT.

3.16

Logical and Physical Containment

a concept that an object can contain another object or be contained within another object

The SM ME addressing scheme is based on the notion of object containment. In the logical CIM schema the ComputerSystem is the container for objects such as CIM Service, ServiceAccessPoint, LogicalDevice, and SystemSpecificCollection. In the physical CIM schema the Rack is the top most physical container. Table 12 and Table 13 describe the containers, the CIM relationship, and the instances they contain.

3.17

Manageability Access Point
MAP

a network-accessible interface for managing a Computer System

A MAP can be instantiated by a Management Process, a Management Processor, a Service Processor, or a Service Process.

3.18

Managed Element
ME

an instance of CIM_ManagedElement

3.19

Managed Element Address

the UFiP to an ME instance

The UFiP may be used as a target of a SM CLP command.

3.20

Managed System Address Space

the fully connected graph of UFiTs that are contained in a specific managed system

The root instance is the ComputerSystem instance that represents the managed system.

3.21

MAP Address Space

the hierarchical graph of the UFiTs contained in the MAP's AdminDomain

Each instance starting at the AdminDomain is a node in the graph. Each supported association forms a link in the graph to another instance node and so on until a terminating instance node is encountered.

3.22

Non-Addressing Association

a CIM association class that is used in an SM profile but is not one of the SM addressing associations that are denoted by a forward slash (/) or a backward slash (\) in the UFiP. See 6.3.

3.23

Physical Element
PE

a tangible element that has a physical manifestation of some sort (as defined in CIM)

296 **3.24**
297 **Root Instance**
298 the top node in a SM address space; the MAP's AdminDomain

299 **3.25**
300 **Server Management**
301 **SM**
302 the ability to remotely administer server-class computer platforms, such as rack-mounted, stand-alone,
303 and blade servers and their respective enclosures

304 **3.26**
305 **System Management Architecture for Server Hardware**
306 **SMASH**
307 an initiative to codify the management interface for heterogeneous servers in the data center,
308 independent of machine state, operating system state, system topology or access method

309 **3.27**
310 **SM Command Line Protocol**
311 **SM CLP**
312 a user friendly command-line protocol to manipulate CIM instances defined by the SM profiles

313 **3.28**
314 **SM profile**
315 a profile defined or referenced by an SM profile specification
316 These profile specifications define CIM profiles for required and recommended instances, properties, and
317 expected behavior to consistently model computer hardware platforms and management domains in
318 order to achieve implementation interoperability. Many of these profiles are listed in DSP0217.

319 **3.29**
320 **Standard User Friendly class Tags**
321 the User Friendly class Tags listed in the UFcT column of Table 1 through Table 11

322 **3.30**
323 **User Friendly class Tag**
324 **UFcT**
325 a short, user friendly synonym for a CIM class name
326 It has the same properties and methods as the CIM class it represents.

327 **3.31**
328 **User Friendly instance Path**
329 **UFiP**
330 a unique path to an instance formed by concatenating the UFiTs of each instance from the root instance
331 to the terminating instance
332 The intervening forward slash (/) or backward slash (\) between each UFiT represents an addressing
333 association.

334 **3.32**
335 **User Friendly instance Suffix**
336 **UFiS**
337 a non-zero positive integer suffix added to a target instance UFcT

338 **3.33**
339 **User Friendly instance Tag**
340 **UFiT**
341 a unique instance tag within the scope of the target instance's containment class
342 A UFiT is created by adding a UFcT to a UFiS that is unique for that UFcT.

343 **3.34**
344 **User Friendly Tag**
345 **UFT**
346 a short, user friendly tag for a CIM class name or instance
347 The two types of UFTs are UFcTs and UFiTs.

4 User Friendly Class Tags

As described previously, a UFcT is designed to provide a short, simple, User Friendly Tag (UFT) for a CIM class. A UFcT is a simple substitute for a CIM class name. The UFcT is the basis for UFiTs according to the containment shown in section 5. Figure 1 and Figure 2 show the logical and physical containment relationships, respectively. Section 6 describes how to create a UFiT from a UFcT based on the instance's containment relationship.

Table 1 through Table 11 define standard UFcTs.

In these tables, the "CIM Class Name" column shows the CIM class name for each UFcT defined in the "UFcT" column. CIM class names are organized according to the containment relationships defined in section 5. The **bold** term in the "UFcT" column is the default UFcT. The non-bold terms in the "UFcT" column list UFcTs that may be used instead of the default UFcT.

Implementations can choose to offer optional OEM-specific UFcTs. Implementations prefix each OEM-specific UFcT with the "OEM" keyword followed by a vendor-unique identification string. The OEM string portion of the prefix uniquely identifies the entity that owns and defines the OEM-specific UFcT. The string includes a copyrighted, trademarked, or otherwise unique name that is owned by the business entity or standards body defining the OEM-specific UFcT.

To enable proper UFcT tag assignments for instances, implementations choose the most derived class offered in the SM Profile (for example, EthernetPort rather than the generic parent NetworkPort). Implementations interpret UFcTs in a case-insensitive fashion.

In Table 1 through Table 11, the "Based On" column gives an explanation of how the UFcT is determined. The value "class" in this column means that the UFcT is based on the CIM class name. If the UFcT is based on the value of a CIM property within the class, the property name and value will be shown in the "Based On" column. If the property is a string, the implementation interprets these strings in a case-insensitive fashion. If the property is a value with a value map, the string provided is interpreted by the implementation as the value map for the corresponding value of that property. Sometimes there is more than one choice for a mapping of a UFcT to a property value. In these cases, the "Comments" column gives some guidelines.

Table 1 – UFcTs for Subclasses of CIM_ManagedElement

CIM Class Name	UFcT	Based On	Comments
CIM_AuthorizedPrivilege	**authorizedpriv**	class	
CIM_ConfigurationCapacity	**configcapacity**	class	
CIM_Location	**location**	class	
CIM_LogRecord	**record**	class	
CIM_Namespace	**namespace**	class	
CIM_PhysicalLocation	**plocation**	class	
CIM_Product	**product**	class	
CIM_RegisteredProfile	**profile**	class	
CIM_RegisteredSubProfile	**subprofile**	class	

376 **Table 2 – UFcTs for Subclasses of CIM_Capabilities**

CIM Class Name	UFcT	Based On	Comments
CIM_MAPCapabilities	**mapcap**	class	
CIM_DeviceSharingCapabilities	**sharingcap**	class	
CIM_PowerManagementCapabilities	**pwrmgtcap**	class	
CIM_SoftwareInstallationServiceCapabilities	**swinstallsvccap**	class	
CIM_StorageCapabilities	**storagecap**	class	
CIM_StorageConfigurationCapabilities	**storagecfgcap**	class	
CIM_DHCPCapabilities	**dhcpcap**	class	

377 **Table 3 – UFcTs for Subclasses of CIM_SettingData**

CIM Class Name	UFcT	Based On	Comments
CIM_BootConfigSetting	**bootcfgsetting**	class	
CIM_BootSourceSetting	**bootsrcsetting**	class	
CIM_BootSettingData	**bootsetting**	class	
CIM_DisplaySetting	**displaysetting**	class	
CIM_MAPSetting	**mapsetting**	class	
CIM_NicSetting	**nicsetting**	class	
CIM_StorageSetting	**storagesetting**	class	
CIM_IPAssignmentSettingData	**ipsettings**	class	
CIM_StaticIPAssignmentSettingData	**staticipsettings**	class	
CIM_DHCPSettingData	**dhcpsettings**	class	
CIM_DNSSettingData	**dnssettings**	class	
CIM_DNSGeneralSettingData	**dnsgeneralsettings**	class	

378 **Table 4 – UFcTs for Subclasses of CIM_Collection**

CIM Class Name	UFcT	Based On	Comments
CIM_Group	**group**	class	
CIM_RedundancySet	**redundancyset**	class	
CIM_SoftwareRepository	**swrepo**	class	
CIM_ConcreteCollection	**concretecollection**	class	
	hdwr	ElementName= “Hardware”	
	capabilities	ElementName = “Capabilities”	
	capacities	ElementName = “Capacities”	
	consoles	ElementName = “Consoles”	
	logs	ElementName = “Logs”	
	profiles	ElementName = “Profiles”	
	privileges	ElementName = “Privileges”	
	products	ElementName = “Products”	
	settings	ElementName = “Settings”	
	sensors	ElementName = “Sensors”	

379 **Table 5 – UFcTs for Subclasses of CIM_LogicalElement**

CIM Class Name	UFcT	Based On	Comments
CIM_SoftwareIdentity	**swid**	class	
CIM_StoragePool	**storagepool**	class	

380 **Table 6 – UFcTs for Subclasses of CIM_EnabledLogicalElement**

CIM Class Name	UFcT	Based On	Comments
CIM_Account	**account**	class	
CIM_ConcreteJob	**job**	class	
CIM_JobQueue	**jobq**	class	
CIM_RecordLog	**log**	class	
CIM_OperatingSystem	**os**	class	

381 **Table 7 – UFcTs for Subclasses of CIM_System**

CIM Class Name	UFcT	Based On	Comments
CIM_AdminDomain	**admin**	class	
CIM_ComputerSystem	**system**	class	
	modular	OtherDedicatedDescriptions= “Modular”	Dedicated= “Other”
	storage	Dedicated= “Storage”	
	router	Dedicated= “Router”	
	switch	Dedicated= “Switch”	
	hub	Dedicated= “Hub”	
	firewall	Dedicated= “Firewall”	
	printserver	Dedicated= “Print”	
	accessserver	Dedicated= “Access Server”	
	ioserver	Dedicated= “I/O”	
	webcache	Dedicated= “Web Caching”	
	management	Dedicated= “Management”	
	blockserver	Dedicated= “Block Server”	
	fileserver	Dedicated= “File Server”	
	mobile	Dedicated= “Mobile User Device”	
	repeater	Dedicated= “Repeater”	
	bridge	Dedicated= “Bridge/Extender”	An implementation may choose either bridge or extender.
	extender	Dedicated= “Bridge/Extender”	An implementation may choose either bridge or extender.
	gateway	Dedicated= “Gateway”	
	storagevlizer	Dedicated= “Storage Virtualizer”	
	medialib	Dedicated= “Media Library”	
	nashead	Dedicated= “NAS Head”	
	nas	Dedicated= “Self-contained NAS”	
	ups	Dedicated= “UPS”	
	ipphone	Dedicated= “IP Phone”	
	map	Dedicated= “Manageability Access Point”	
	sp	Dedicated= “Management Controller”	
	chassismgr	Dedicated= “Chassis Manager”	

382 **Table 8 – UFcTs for Subclasses of CIM_LogicalDevice**

CIM Class Name	UFcT	Based On	Comments
CIM_AlarmDevice	**alarm**	class	
CIM_Battery	**battery**	class	
CIM_CDROMDrive	**cd**	class	
CIM_CoolingDevice	**cooling**	class	
CIM_DAPort	**daport**	class	
CIM_DiskDrive	**diskdrive**	class	
CIM_DisketteDrive	**floppy**	class	
CIM_DiskPartition	**diskpartition**	class	
CIM_Display	**display**	class	
CIM_DVDDrive	**dvd**	class	
CIM_Fan	**fan**	class	
CIM_FCPort	**fcport**	class	
CIM_HeatPipe	**heatpipe**	class	
CIM_IBPort	**ibport**	class	
CIM_Keyboard	**keyboard**	class	
CIM_LogicalDisk	**disk**	class	
CIM_LogicalModule	**logicalmodule**	class	
	devicetray	LogicalModuleType = “Device Tray”	
	linecard	LogicalModuleType = “Line Card”	
	blademodule	LogicalModuleType = “Blade”	
CIM_LogicalPort	**logicalport**	class	
CIM_MediaAccessDevice	**mediaaccess**	class	
CIM_Memory	**memory**	class	
CIM_Modem	**modem**	class	
CIM_NetworkPort	**netport**	class	
CIM_WirelessPort	**wifiport**	class	
CIM_EthernetPort	**enetport**	class	
CIM_PCIDevice	**pcidev**	class	
CIM_PCIBridge	**pcibridge**	class	
CIM_PointingDevice	**pointer**	class	
	mouse	PointingType= “Mouse”	
	trackball	PointingType= “Track Ball”	
	touchpad	PointingType= “Touch Pad”	
	touchscreen	PointingType= “Touch Screen”	

CIM Class Name	UFcT	Based On	Comments
CIM_PortController	**portctlr**	class	
	nic	ControllerType= “Ethernet”	
	hca	ControllerType= “IB”	
	tca	ControllerType= “IB”	
	hba	ControllerType= “FC”	
CIM_PowerSupply	**pwrsupply**	class	
CIM_Printer	**printer**	class	
CIM_Processor	**cpu**	class	
CIM_Refrigeration	**refrigeration**	class	
CIM_SCSIProtocolController	**scsiprotctlr**	class	
CIM_Sensor	**sensor**	class	
	currentsensor	SensorType= “Current”	
	tachsensor	SensorType= “Tachometer”	
	tempsensor	SensorType= “Temperature”	
	voltsensor	SensorType= “Voltage”	
	countersensor	SensorType= “Counter”	
	switchsensor	SensorType= “Switch”	
	locksensor	SensorType= “Lock”	
	humiditysensor	SensorType= “Humidity”	
	airsensor	SensorType= “Air Flow”	
	presencesensor	SensorType= “Presence”	
	smokesensor	SensorType= “Smoke Detection”	
CIM_NumericSensor	**numsensor**	class	
	ncurrentsensor	SensorType= “Current”	
	ntachsensor	SensorType= “Tachometer”	
	ntempsensor	SensorType= “Temperature”	
	nvoltsensor	SensorType= “Voltage”	
	ncountersensor	SensorType= “Counter”	
	nswitchsensor	SensorType= “Switch”	
	nlocksensor	SensorType= “Lock”	
	nhumiditysensor	SensorType= “Humidity”	
	nairsensor	SensorType= “Air Flow”	
	npresencesensor	SensorType= “Presence”	
	nsmokesensor	SensorType= “Smoke Detection”	
CIM_SPIPort	**spiport**	class	
CIM_StorageVolume	**storagevol**	class	
CIM_StorageExtent	**storageext**	class	
CIM_SerialPort	**serialport**	class	
CIM_TapeDrive	**tapedrive**	class	

CIM Class Name	UFcT	Based On	Comments
CIM_USBPort	**usbport**	class	
CIM_WatchDog	**watchdog**	class	
CIM_PortController	**portctrl**	class	

383 **Table 9 – UFcTs for Subclasses of CIM_Service**

CIM Class Name	UFcT	Based On	Comments
CIM_BootService	**bootsvc**	class	
CIM_CLPService	**clpsvc**	class	
CIM_IPConfigurationService	**ipcfgsvc**	class	
CIM_PowerManagementService	**pwrmgtsvc**	class	
CIM_SharedDeviceManagementService	**shareddevicesvc**	class	
CIM_SoftwareInstallationService	**swinstallsvc**	class	
CIM_SSHService	**sshsvc**	class	
CIM_StorageConfigurationService	**storagecfgsvc**	class	
CIM_TelnetService	**telnetsvc**	class	
CIM_TextRedirectionService	**textredirectsvc**	class	
CIM_TimeService	**timesvc**	class	

384 **Table 10 – UFcTs for Subclasses of CIM_ServiceAccessPoint**

CIM Class Name	UFcT	Based On	Comments
CIM_ProtocolEndpoint	**protoendpt**	class	
CIM_IPProtocolEndpoint	**ipendpt**	class	
CIM_DHCPProtocolEndpoint	**dhcpendpt**	class	
CIM_DNSProtocolEndpoint	**dnsendpt**	class	
CIM_RemoteServiceAccessPoint	**remotesap**	class	
	dnsserver	AccessContext= "DNS Server"	
	dhcpserver	AccessContext= "DHCP Server"	
	gateway	AccessContext= "Default Gateway"	
CIM_LANEndpoint	lanendpt	class	
CIM_RemotePort	**remoteport**	class	
CIM_SCSIProtocolEndPoint	**scsiendpt**	class	
CIM_ServiceAccessURI	**serviceuri**	class	
CIM_TextRedirectionServiceAccessPoint	**textredirectsap**	class	

385 **Table 11 – UFcTs for Subclasses of CIM_PhysicalElement**

CIM Class Name	UFcT	Based On	Comments
CIM_PhysicalPackage	**pkg**	class	
	bladepkg	PackageType= “Blade”	
	bladexpkg	PackageType= “Blade Expansion”	
	diskpkg	PackageType= “Storage Media Package”	
	fanpkg	PackageType= “Fan”	
	pwrpkg	PackageType= “Power Supply”	
	rackpkg	PackageType= “Rack”	
	chassispkg	PackageType= “Chassis/Frame”	If the package is a chassis rather than a frame, choose chassispkg.
	framepkg	PackageType= “Chassis/Frame”	If the package is a frame rather than a chassis, choose framepkg.
	backplanepkg	PackageType= “Crossconnect/Backplane”	
	sensorpkg	PackageType= “Sensor”	
	modulepkg	PackageType= “Module/Card”	If the package is a module rather than a card, choose modulepkg.
	cardpkg	PackageType= “Module/Card”	If the package is a card rather than a module, choose cardpkg.
	batterypkg	PackageType= “Battery”	
	cpupkg	PackageType= “Processor”	
	memorypkg	PackageType= “Memory”	
	storagepkg	PackageType= “Storage Media Package”	
	pwrsrcpkg	PackageType= “Power Source/Generator”	
CIM_PhysicalFrame	**frame**	class	
CIM_Rack	**rack**	class	
CIM_Chassis	**chassis**	class	
	laptop	ChassisPackageType= “LapTop”	
	desktop	ChassisPackageType= “Desktop”	
	tower	ChassisPackageType= “Tower”	
	storagechas	ChassisPackageType= “Storage Chassis”	
	notebook	ChassisPackageType= “Notebook”	
	mainchassis	ChassisPackageType= “Main	

CIM Class Name	UFcT	Based On	Comments
		System Chassis”	
	expansion	ChassisPackageType= “Bus Expansion Chassis”	
	peripheralchassis	ChassisPackageType= “Peripheral Chassis”	
	subchassis	ChassisPackageType= “SubChassis”	
CIM_Card	**card**	class	
CIM_SystemBusCard	**buscard**	class	
	pcicard	BusType= “PCI”	
	eisacard	BusType= “EISA”	
	vesacard	BusType= “VESA”	
	pcmciacard	BusType= “PCMCIA”	The UFcT does not discriminate among PCMCIA bus card types.
		BusType= “PCMCIA Type I”	
		BusType= “PCMCIA Type II”	
		BusType= “PCMCIA Type III”	
	accesscard	BusType= “Access.bus”	
	nubuscard	BusType= “NuBus”	
	agpcard	BusType= “AGP”	
	vmecard	BusType= “VME Bus”	
	pccard	BusType= “PC-98”, “PC-98-Hireso”, “PC-H98”, PC-98Note”, “PC-98Full”	
	pcixcard	BusType= “PCI-X”	
	pciecard	BusType= “PCI-E”	
	sbuscard	BusType= “Sbus IEEE 1396-1993 32 bit”, “Sbus IEEE 1396-1993 64 bit”	
	isacard	BusType= “ISA”	
	mcacard	BusType= “MCA”	
	giocard	BusType= “GIO”	
	xiocard	BusType= “XIO”	
	hiocard	BusType= “HIO”	
	pmccard	BusType= “PMC”	
	ibcard	BusType= “Infiniband”	
CIM_PhysicalComponent	**component**	class	
CIM_Chip	**chip**	class	
	propchip	FormFactor= “Proprietary Chip”	
	sip	FormFactor= “SIP”	
	dip	FormFactor= “DIP”	
	zip	FormFactor= “ZIP”	
	soj	FormFactor= “SOJ”	
	simm	FormFactor= “SIMM”	
	dimm	FormFactor= “DIMM”	

CIM Class Name	UFcT	Based On	Comments
	tsop	FormFactor= “TSOP”	
	pga	FormFactor= “PGA”	
	rimm	FormFactor= “RIMM”	
	sodimm	FormFactor= “SODIMM”	
	srimm	FormFactor= “SRIMM”	
	smd	FormFactor= “SMD”	
	ssmp	FormFactor= “SSMP”	
	qfp	FormFactor= “QFP”	
	tqfp	FormFactor= “TQFP”	
	soic	FormFactor= “SOIC"	
	lc	FormFactor= “LCC”	
	plcc	FormFactor= “PLCC”	
	bga	FormFactor= “BGA”	
	fpbga	FormFactor= “FPBGA”	
	lga	FormFactor= “LGA”	
CIM_PhysicalMemory	**pmem**	class	
	ram	MemoryType= “RAM”	
	dram	MemoryType= “DRAM”	
	synchdram	MemoryType= “Synchronous DRAM”	
	cache	MemoryType= “Cache DRAM”	
	edo	MemoryType= “EDO”	
	edram	MemoryType= “EDRAM”	
	vram	MemoryType= “VRAM”	
	sram	MemoryType= “SRAM”	
	flash	MemoryType= “Flash”	
	eeprom	MemoryType= “EEPROM”	
	eprom	MemoryType= “EPROM”	
	cdram	MemoryType= “CDRAM”	
	sdram	MemoryType= “SDRAM”	
	sgram	MemoryType= “SGRAM”	
	rdram	MemoryType= “RDRAM”	
	ddr	MemoryType= “DDR”	
	bram	MemoryType= “BRAM”	
CIM_PhysicalConnector	**connector**	class	
CIM_Slot	**slot**	class	

5 SM Containment Hierarchy

This section describes the SM CIM-based containment hierarchy that forms the foundation of the SM ME addressing standard.

5.1 General Concepts

The physical containment hierarchy defines how PhysicalElement classes are addressed. The logical containment hierarchy defines how LogicalElement classes are addressed. The SM containment hierarchy is based on CIM Schema 2.10 and will be revised to accommodate later CIM Schema versions as required.

CIM divides its schema into two realms: logical objects and physical objects. All logical objects derive from the LogicalElement class. All physical objects derive from the PhysicalElement class.

In CIM, all aspects of an object that are not related to its three-dimensional, concrete real world existence are modeled as LogicalElement instances. This includes devices. A CIM_PhysicalElement (PE) is an object that occupies space and can be touched. A LogicalElement, on the other hand, does not occupy space and cannot be touched. This label might seem counterintuitive for LogicalElement subclasses such as ComputerSystem and LogicalDevice. The ComputerSystem is the LogicalElement that comprises all the functionality of a system — not just the hardware, but the software, firmware, processes, file systems, Internet addresses, logs, and so on. In a single-chassis system one can imagine that one is touching the entire computer system, but in reality one is touching only the form factor that the computer is enclosed within. Similarly, one may argue that a device is a tangible thing. One can touch a disk drive. As in the previous example, the disk drive is the *physical* form factor (a PhysicalPackage) that encloses the *logical* aspects of a disk, such as storage volumes, disk partitions, and memory blocks. In CIM, only the form factor is a PhysicalPackage, and all the rest, including the logical entity that describes the kind of disk drive (for example, floppy, CD, DVD), are LogicalDevice instances.

The SM ME addressing scheme is based on the hierarchical pattern of associations between CIM objects. A UFiP is created by identifying the specific object (class or instance) tags (UFiTs) to a leaf object starting from the root object (AdminDomain). SM ME addressing uses a forward slash (/) or a backward slash (\) between object terms to signify that an addressing association exists between the terms. The SM containment hierarchies define the addressing associations. The addressing associations are listed in 5.5. The SM ME addressing containment relationships are listed in Table 12 and Table 13. These relationships are stated more formally in 6.3.

Another concept worth noting is the difference between a class and an instance of a class. A class is the template that one uses to create an instance. The class defines the nature of the properties and methods: data type, size, range, signature (input, output), and so on. An instance is a unique instantiation of the class template with specific information for each variable term. So, when one uses a class term (UFcT), such as cpu or disk, one is addressing a particular class. When one uses an instance term (UFiT), one is addressing a particular object. For example, cpu1 or disk3 contain specific information for one particular CPU or disk.

5.2 Logical Containment

The root of the SM MAP address space is the AdminDomain. The term “root” in this context denotes the highest order container in which the MAP stores information. Each class in the hierarchy is considered to be a container of its own information, including association classes that contain information regarding the classes they associate.

The ComputerSystem instance is the root of its address space. That is, all the information about the ManagedElement instances within a single computer system is contained within the ComputerSystem instance through associations to the contained instances.

The UML diagram in Figure 1 defines the SM logical containment hierarchy. At the top of the diagram is the AdminDomain class. For each ComputerSystem instance managed by a MAP, a <*SystemComponent*> association exists between the AdminDomain instance and the ComputerSystem instance. From here the client can traverse intervening addressing associations to address SM MEs. It is the job of the parser to validate that each UFiP is valid according to the grammar explained in 6.2.

In Figure 1, we see that SM logical containment hierarchy provides the following associations to address logical entities within an instance of the ComputerSystem class and to define their containment relationship. The ← notation denotes the direction of containment. The container is on the left of the arrow head and is read as “contains”. The addressing association is in <*italics*>.

NOTE: This is not an exhaustive list, and the information in Figure 1 and Figure 2 is not exhaustive.

ComputerSystem ← <*HostedService*> ← Service

ComputerSystem ← <*HostedAccessPoint*> ← ServiceAccessPoint

ComputerSystem ← <*OwningCollectionElement*> ← ConcreteCollection

ComputerSystem ← <*HostedPool*> ← StoragePool

ComputerSystem ← <*SystemComponent*> ← ComputerSystem

ComputerSystem ← <*SystemDevice*> ← LogicalDevice

The complete lists of allowable managed elements are in Table 1 through Table 11 and section 6.

Note that the cardinality in the CIM Schema may be different than what is represented in Figure 1 and Figure 2. In order to enforce containment, this specification restricts the cardinality of the aggregations to what is shown in Figure 1 and Figure 2.

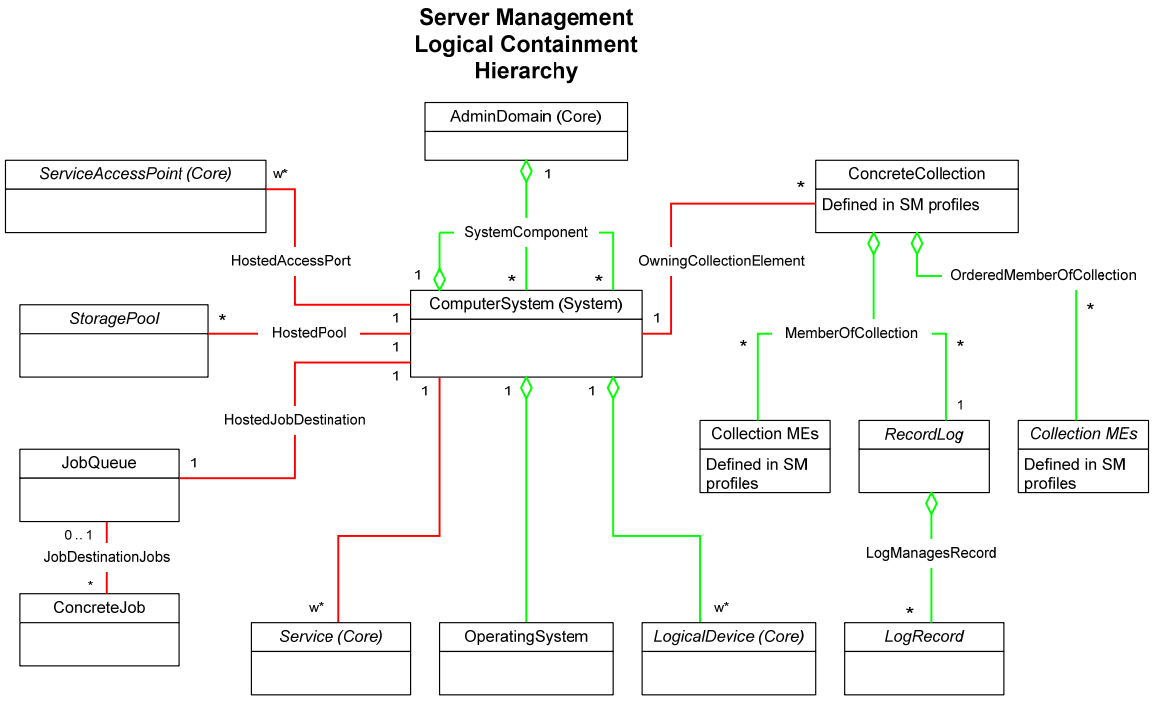

Figure 1 – SM Logical Containment UML Diagram

453 The relationships described in the previous paragraph can be expressed in the following hierarchical
454 notation:

455 AdminDomain/ComputerSystem

456 AdminDomain/ComputerSystem/Service

457 AdminDomain/ComputerSystem/ServiceAccessPoint

458 AdminDomain/ComputerSystem/ConcreteCollection

459 AdminDomain/ComputerSystem/ConcreteCollection/ME

460 AdminDomain/ComputerSystem/ConcreteCollection/RecordLog

461 AdminDomain/ComputerSystem/ConcreteCollection/RecordLog/LogRecord

462 AdminDomain/ComputerSystem/LogicalDevice

463 AdminDomain/ComputerSystem/ComputerSystem/LogicalDevice

464 In the preceding list, the forward slash (/) and backward slash (\) notation indicates the intervening
465 addressing association. Section 6.1 describes how a UFiT is created from a UFcT. The resultant UFiTs
466 can then be substituted for the CIM class names in the preceding list to form distinct SM ME instance
467 addresses. Multiple addressing associations may associate an ME to its containers. The implementation
468 selects exactly one of these addressing associations to use for addressing all instances of the ME.

469 In Figure 1, CIM defines that a single AdminDomain instance may contain 0 to *n* ComputerSystem
470 instances and 0 to *n* Collection instances. *A* ComputerSystem instance contained in the AdminDomain
471 instance may contain 0 to *n* Service instances; 0 to *n* ServiceAccessPoint instances; 0 to *n* Collection
472 instances; 0 to *n* LogicalDevice instances; 0 to *n* StoragePool instances; and 0 to *n* contained
473 ComputerSystem instances that also may contain 0 to n LogicalDevice instances, Service instances, and
474 so on. This observation forms the basis of the addressing rules defined in section 6.

475 The ConcreteCollection class represents all the subclasses of that class that appear in the SM profiles,
476 including SoftwareRepository, RedundancySet, Group, and ConcreteCollection. The term “Collection
477 MEs” associated to the Collection through MemberOfCollection or OrderedMemberOfCollection
478 represents the MEs that are contained in the Collection (or a subclass) instance.

479 Similarly, ComputerSystem, ServiceAccessPoint, LogicalDevice, and Service represent all the subclasses
480 of these classes that appear in the SM profiles.

481 5.3 Physical Containment

482 The example in Figure 2 shows the SM physical containment hierarchy and the association hierarchy that
483 defines the SM ME addressing scheme and the containment associations for server hardware managed
484 in the MAP’s AdminDomain class.

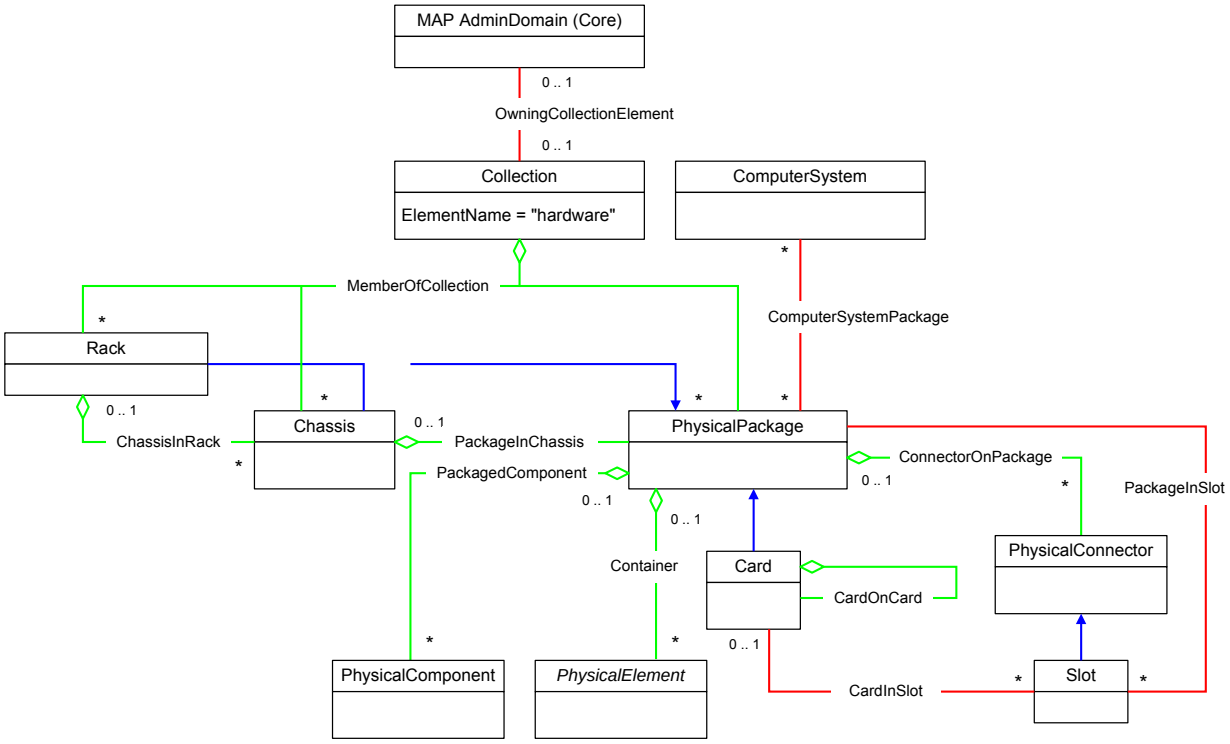

Figure 2 – SM Physical Containment UML Diagram

CIM defines several aggregation associations for PEs. SM addressing leverages the rules of the associations defined in Figure 2 to uniquely address any PE. Figure 2 shows that the MAP AdminDomain contains a single ConcreteCollection instance with the ElementName property set to “hardware”. Through the <*MemberOfCollection*> association, this collection contains the topmost PhysicalPackage instances of a ComputerSystem instance that the MAP manages. If this is a Rack object, then it is the top of the hierarchy that contains all the PEs in that Rack. A Chassis within this Rack would be related with a <*ChassisInRack*> association and the PhysicalPackage instance in each Chassis instance would be related with a <*PackageInChassis*> association. Similarly, if the topmost PhysicalPackage instance of a ComputerSystem managed by the MAP is a Chassis or PhysicalPackage, the enclosed PEs would use the associations described in Figure 2.

A ComputerSystem instance may be composed of hardware components that are contained in a single form factor (Chassis) or components that span multiple Racks. Figure 2 shows that the ComputerSystem is associated to instances of its topmost PhysicalPackage instances through the <*ComputerSystemPackage*> association (DPS0217). Similarly, Figure 2 shows that each PE in a ComputerSystem instance can be associated with one or more specific LogicalDevice instances using the <*Realizes*> association. The PhysicalElement inheritance hierarchy is shown in Figure 3. For the complete PhysicalElement hierarchy, refer to the CIM Schema.

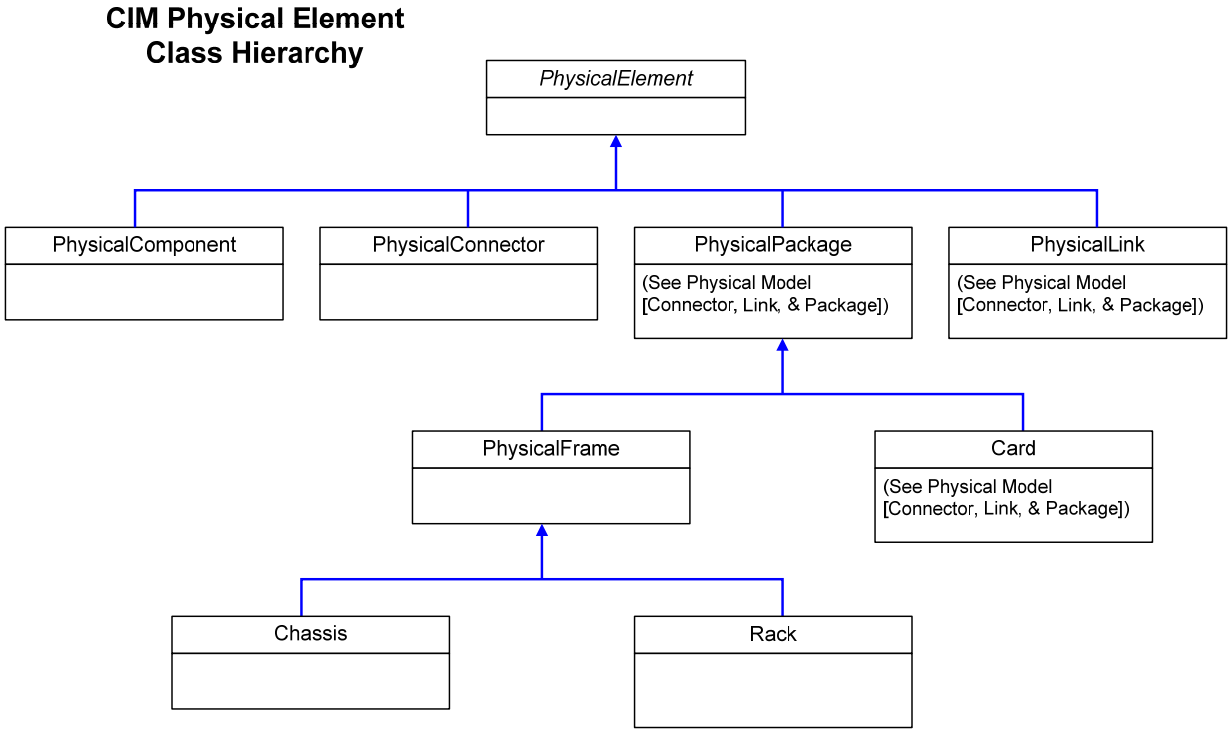

Figure 3 – CIM PhysicalElement Class Hierarchy

The PhysicalElement inheritance hierarchy defines that a PE may be an instance of PhysicalPackage (frame, rack, and chassis), Component, Connector, or a PhysicalLink. Because a Chassis is a PhysicalPackage it may be substituted for a PhysicalPackage in the physical containment diagram shown in Figure 2, adding *n* levels of recursion. A PhysicalPackage may also contain any PE, adding another level of recursion. Thus, the potential hardware containment relationship hierarchy is *n* deep, which is complex enough to model any arbitrary computer hardware topology. The SM profile specifications provide guidelines for vendors on how to model their hardware platforms to conform to the containment hierarchy. Vendors decide which components in their systems will be manageable by creating the appropriate ME instances. The rules for ME containment are discussed in sections 5.4 and 6.

Figure 2 helps to illustrate how a MAP addresses the hardware components of the ComputerSystem instances that it manages. Figure 2 shows the MAP AdminDomain top-down containment relationships as follows:

AdminDomain ← <*OwningCollectionElement*> ← ConcreteCollection ← <*MemberOfCollection*> ← PhysicalPackage

Rack ← <*ChassisInRack*> ← Chassis ← <*PackageInChassis*> ← PhysicalPackage

← <*PackagedComponent*> ← PhysicalComponent

or

← <*ConnectorOnPackage*> ← PhysicalConnector

or

← <*Container*> ← PhysicalElement

or

Chassis ← <*PackageInChassis*> ← PhysicalPackage

528 ← <*PackagedComponent*> ← PhysicalComponent
529 or
530 ← <*ConnectorOnPackage*> ← PhysicalConnector
531 or
532 ← <*Container*> ← PhysicalElement
533 or
534 PhysicalPackage ← <*PackagedComponent*> ← PhysicalComponent
535 or
536 ← <*ConnectorOnPackage*> ← PhysicalConnector
537 or
538 ← <*Container*> ←PhysicalElement

539 By definition, the MAP AdminDomain contains a single ConcreteCollection instance that contains 0 to *n*
540 PhysicalPackage instances (Rack, Chassis or PhysicalPackage) that represent the topmost physical
541 containers of the ComputerSystem instances that the MAP is managing. A single Rack instance contains
542 0 to *n* Chassis instances. A single Chassis instance contains 0 to *n* PhysicalPackage instances. A single
543 PhysicalPackage instance contains 0 to *n* PhysicalComponent, PhysicalConnector, or PhysicalElement
544 instances.

545 The SM containment rules in section 6 guarantee that contained elements are unique. The SM physical
546 containment hierarchy associations prescribe how containers are ordered. The specific association is
547 never ambiguous because it is explicit in the instance. CIM uses the term REF to denote a data type that
548 is a Reference to an instance. A REF is an object pointer that is used to link two objects in an association
549 (CIM Schema 2.10). If a Chassis is in a Rack, it will be the Chassis REF in a <*ChassisInRack*>
550 association**.** If a Chassis is in a PhysicalPackage it will be the PE REF in a <*Container*> association. That
551 is, the Chassis is either immediately contained in the Rack class or immediately contained within some
552 PhysicalPackage subclass (for example, a Frame or another Chassis). The SM physical containment
553 hierarchy shows that only a PhysicalConnector instance can have a <*ConnectorOnPackage*> association
554 to a PhysicalPackage instance and only a PhysicalComponent instance can have a
555 <*PackagedComponent*> association to a PhysicalPackage instance. Any PE may be the PE REF in a
556 <*Container*> association to a PhysicalPackage instance. However, the PE in the <*Container*> association
557 will not be a PartComponent REF in another containment association.

558 The MAP AdminDomain addressing associations and containment relationships are expressed in the
559 following notation, where the forward slash (/) and backward slash (\) notation indicates the intervening
560 association defined in Figure 2:

561 AdminDomain/ConcreteCollection/Rack/Chassis/PhysicalPackage

562 AdminDomain/ConcreteCollection/Rack/Chassis/PhysicalPackage/PhysicalConnector

563 AdminDomain/ConcreteCollection/Rack/Chassis/PhysicalPackage/PhysicalComponent

564 AdminDomain/ConcreteCollection/Rack/Chassis/PhysicalPackage/PhysicalElement

565 AdminDomain/ConcreteCollection/Chassis/PhysicalPackage

566 AdminDomain/ConcreteCollection/Chassis/PhysicalPackage/PhysicalConnector

567 AdminDomain/ConcreteCollection/Chassis/PhysicalPackage/PhysicalComponent

568 AdminDomain/ConcreteCollection/Chassis/PhysicalPackage/PhysicalElement

569 AdminDomain/ConcreteCollection/PhysicalPackage

570 AdminDomain/ConcreteCollection/PhysicalPackage/PhysicalConnector

571 AdminDomain/ConcreteCollection/PhysicalPackage/PhysicalComponent

572 AdminDomain/ConcreteCollection/PhysicalPackage/PhysicalElement

573 Note that the physical containment hierarchy can be arbitrarily deep based on the hardware system being
574 represented.

575 An implementation may choose not to implement instances of every PE class; therefore, in those
576 implementations, some terms in the address path may not appear. Required associations include one of
577 the following, depending on the type of the topmost PhysicalPackage being managed:

578 AdminDomain/ConcreteCollection/Rack

579 AdminDomain/ConcreteCollection/Chassis

580 AdminDomain/ConcreteCollection/PhysicalPackage

581 MAP implementations define which PEs are available to be managed by creating instances of the PE
582 according to the SM profiles. Section 6.1 describes how a UFiT is created from a UFcT. The resultant
583 UFiTs can then be substituted for the CIM class names in the physical containment hierarchy to form
584 distinct SM PE instance addresses.

585 5.4 SM ME Addressing Hierarchy

586 In summary, the SM logical containment hierarchy uses CIM logical containment associations to address
587 the Service, ServiceAccessPoint, Collection, ComputerSystem, StoragePool, and LogicalDevice
588 instances in a ComputerSystem instance in the MAP's AdminDomain instance. The SM physical
589 containment hierarchy defines how to address physical components in the MAP's AdminDomain. The
590 addressing rules (section 6.2) define how to create a UFiT from a UFcT so that it is unique within the
591 managed ComputerSystem.

592 The diagram in Figure 4 combines the components of the logical and physical containment hierarchies to
593 define the SM ME addressing hierarchy.

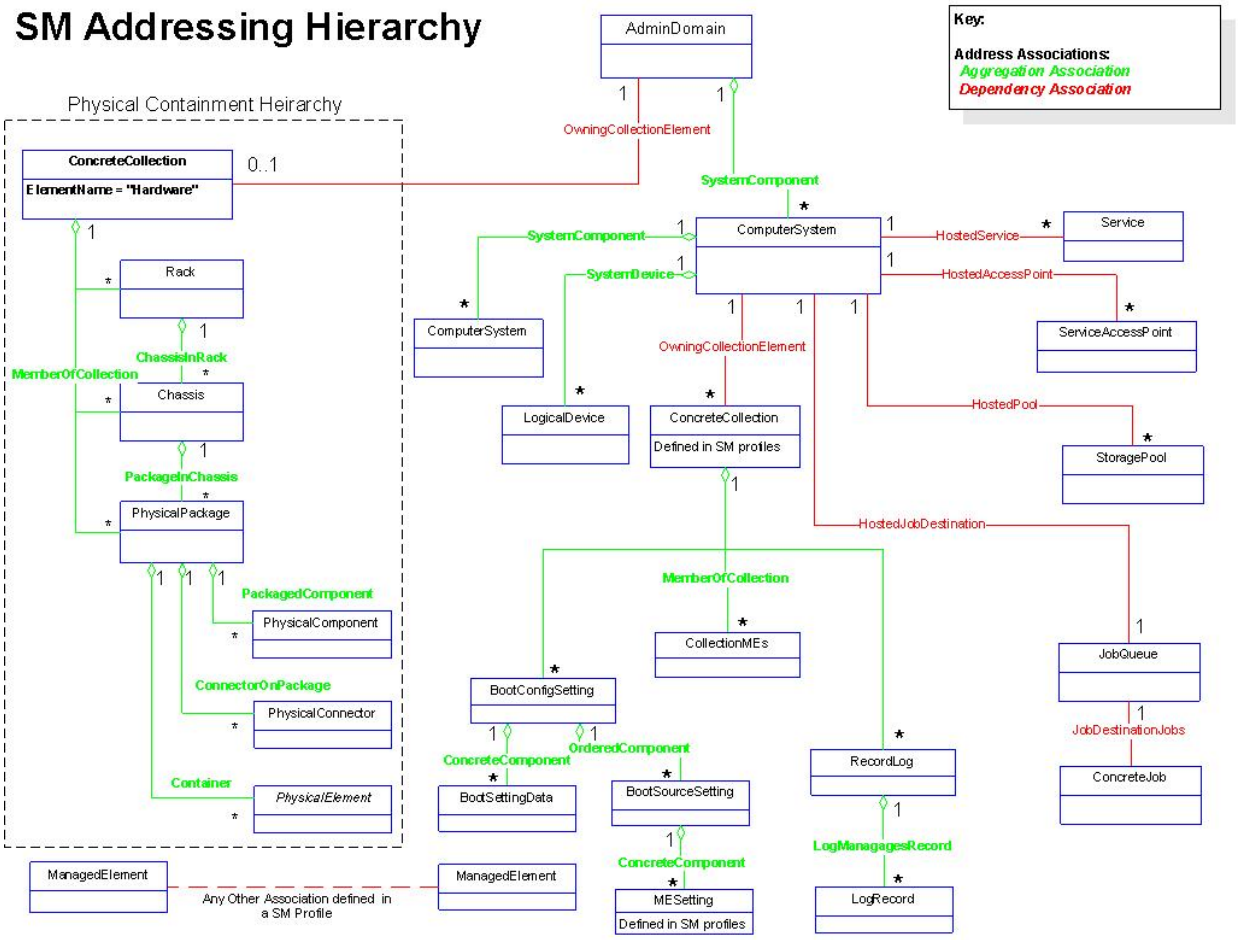

Figure 4 – High-Level View of the SM ME Addressing Hierarchy

Figure 4 depicts a high-level example of the SM ME addressing hierarchy; it is not intended to be an exhaustive list of addressing associations. Section 6.3 presents all the valid addressing terms. Figure 4 shows a single hierarchical address space rooted at the AdminDomain. An address path is one in which each term has the appropriate intervening addressing association and role. This is explained in detail in 6.2. Figure 4 shows the addressing associations between the path terms that must exist at each level until the last (leaf) term is reached.

Table 12 lists the SM physical containment associations based on the physical containment hierarchy. These associations are used in section 6 to derive unique instance tags for SM PEs.

604 **Table 12 – SM Hardware Containment Relationships**

Container Class	Addressing Association Class	Contained Class
CIM_AdminDomain	CIM_*OwningCollectionElement*	CIM_ConcreteCollection
CIM_ConcreteCollection	CIM_*MemberOfCollection*	CIM_PhysicalPackage
CIM_Rack	CIM_*ChassisInRack*	CIM_Chassis
CIM_Chassis	CIM_*PackageInChassis*	CIM_PhysicalPackage
CIM_PhysicalPackage	CIM_*Container*	CIM_PhysicalElement
CIM_PhysicalPackage	CIM_*PackagedComponent*	CIM_PhysicalComponent
CIM_PhysicalPackage	CIM_*ConnectorOnPackage*	CIM_PhysicalConnector
CIM_PhysicalConnector	CIM_*PackageInConnector*	CIM_PhysicalPackage
CIM_Slot	CIM_*PackageInSlot*	CIM_PhysicalPackage
CIM_Slot	CIM_*CardInSlot*	CIM_Card
CIM_Card	CIM_*CardOnCard*	CIM_Card

605 Table 13 lists the SM logical containment associations based on the logical containment hierarchy. These
606 associations are used in section 6 to derive unique instance tags for SM logical MEs.

607 **Table 13 – SM Logical Containment Relationships**

Container Class	Addressing Association Class	Contained class
CIM_AdminDomain	CIM_*SystemComponent*	CIM_ComputerSystem
CIM_ComputerSystem	CIM_*SystemComponent*	CIM_ComputerSystem
CIM_ComputerSystem	CIM_*SystemDevice*	CIM_LogicalDevice
CIM_ComputerSystem	CIM_*HostedService*	CIM_Service
CIM_ComputerSystem	CIM_*HostedAccessPoint*	CIM_ServiceAccessPoint
CIM_ComputerSystem	CIM_*OwningCollectionElement*	CIM_ConcreteCollection
CIM_ComputerSystem	CIM_*HostedPool*	CIM_StoragePool
CIM_ConcreteCollection	CIM_*MemberOfCollection*	CIM_ManagedElement
CIM_ConcreteCollection	CIM_*OrderedMemberOfCollection*	CIM_ManagedElement
CIM_BootConfigSetting	CIM_*ConcreteComponent*	CIM_SettingData
CIM_BootConfigSetting	CIM_*OrderedComponent*	CIM_BootSourceSetting
CIM_BootSourceSetting	CIM_*ConcreteComponent*	CIM_SettingData
CIM_RecordLog	CIM_*LogManagesRecord*	CIM_LogRecord
CIM_ComputerSystem	CIM_*HostedJobQueue*	CIM_JobQueue
CIM_JobQueue	CIM_*JobDestinationJobs*	CIM_ConcreteJob
CIM_ProtocolEndpoint	*CIM_BindsTo*	CIM_ProtocolEndpoint
CIM_ProtocolEndpoint	*CIM_RemoteAccessAvailableToElement*	CIM_RemoteServiceAccessPoint
CIM_EthernetPort	*CIM_PortImplementsEndpoint*	CIM_LANEndpoint
CIM_IPAssignmentSettingData	*CIM_OrderedComponent*	CIM_IPAssignmentSettingData

5.5 Addressing Associations

This specification not only defines a way of addressing instances of classes but also instances of associations. The SM ME addressing grammar defines the following syntax to address CIM association instances:

"UFiP=> <TargetAssoc> "

"UFiP=> <TargetAssoc> =>UFiP"

The first form targets all the associations of type <TargetAssoc> that reference the instance specified in the preceding UFiP, and the second form targets the association instance that links the UFiP on each side of the expression.

The following example may help explain the use of this feature. The example shown in Figure 1 has these associations:

Admin Domain ← <*SystemComponent*> ← ComputerSystem ← <*SystemDevice*> ← LogicalDevice

Using the backward and forward slashes for the associations yields the following:

AdminDomain/ComputerSystem/LogicalDevice

If the LogicalDevice were a Processor, a possible UFiP for the previous example could be as follows:

admin1/system1/cpu1

To address all instances of the *SystemDevice* association that were associated with this instance of AdminDomain/ComputerSystem, the following syntax could be used:

admin1/system1=>systemdevice

To address the specific instance of *SystemDevice* that is associated with this instance of LogicalDevice under this AdminDomain/ComputerSystem, the following syntax could be used:

admin1/system1=>systemdevice=>admin1/system1/cpu1

6 SM ME Addressing

Section 6.1 describes how UFiTs are created. Section 6.2 defines the addressing rules to which implementations shall adhere. Section 6.3 shows the productions for the SM ME addressing grammar. This grammar provides the basis to develop algorithms to programmatically parse a UFiP.

6.1 Instance Tagging

The User Friendly instance Tag (UFiT) is constructed by adding an instance suffix to the UFcT (**default** or specific). The instance suffix is a positive non-zero integer and is unique within its container for any given UFcT. It is recommended that suffixes within a container be numbered starting at one and increase in value sequentially.

A unique UFiT within a container addresses a single ME instance. ANNEX A shows several examples of how to create SM UFiTs and UFiPs based on an instance's UFcT and its container.

6.2 Addressing Rules

Table 14 defines the rules that implementations shall obey to support SM ME addressing.

Table 14 – SM ME Addressing Rules

Rule	Description
1	Implementations shall support UFcTs. Implementations shall support the **default** UFcT for each instance of a class implemented as defined in the profile being implemented. Implementations shall derive the UFcT from the implemented class. Implementations shall use the standard UFcTs from Table 1 through Table 11 for the instantiated class. Implementations may use any of the specific UFcTs listed in Table 1 through Table 11 for the instantiated class. If an implementation uses the **default** UFcT**,** then the property that defines the specific UFcT may be null. If an implementation uses a specific UFcT based on a property, that property value shall not be null and shall agree with the property value listed for the specific UFcT in the "Based On" column of the Standard User Friendly class Tags tables (Table 1 through Table 11). If the property is a string, the implementation shall interpret the value of the property in a case-insensitive fashion. If the property is a value-mapped property, the interpretation shall interpret the string specified as the ValueMap for the corresponding value. If the optional, specific UFcT listed in Table 1 through Table 11 is based on a value that is an element in an array property, the implementation shall use the value of the first element in the array as the selector for the optional, specific UFcT. Implementations shall interpret all UFcTs and UFiTs in a case-insensitive fashion.
2	Implementations shall support the following rules for creating instance tags from SM UFcT class root and container class defined in 6.1: User Friendly instance Tag, UFiT, shall be constructed by adding an instance suffix to the UFcT (**default** or specific). The instance suffix shall be a positive non-zero integer. The instance suffix shall be unique within its container for any given UFcT. The instance suffix is not required to be a sequential enumeration. The instance suffix is not required to begin at any given value for any container.
3	Implementations' ME addresses shall comply with the syntax specified by the SM ME addressing grammar

Rule	Description
	as specified in 6.3.
4	Implementations shall define ME addresses according to the addressing associations defined in the SM containment hierarchy and SM profiles: Implementations shall provide exactly one UFiP from the AdminDomain to every ME. Implementations shall guarantee that a UFiP identifies exactly one ME instance. Implementations shall choose at most one addressing association to address an instance of an ME. Implementations shall use one of the addressing associations listed in Table 12 or Table 13 as the association used to define the relationship between the instance of the ME and its containing ME.
5	Implementations shall not allow UFiTs to be modified by the client.
6	Implementations shall guarantee that the AdminDomain is the highest order node of the MAP's address space.
7	Implementations may provide OEM-specific extensions to the standard UFcTs. The meaning of any "OEM" prefixed UFcT is outside the scope of this specification. Implementations that support OEM-specific extensions shall adhere to the following rules: Implementations shall prefix each OEM-specific UFcT with the keyword "OEM" followed by a vendor-unique identification string. The OEM string portion of the prefix shall uniquely identify the entity that owns and defines the OEM-specific UFcT. The string shall include a copyrighted, trademarked, or otherwise unique name that is owned by the business entity or standards body defining the OEM-specific UFcT.
8	Implementations are not required to persist the UFiT assignments over the life of a configuration. Implementations may provide a mechanism for the client to force UFiTs to be recycled. Implementations shall adhere to the following rules: Implementations shall define the conditions under which UFiTs are destroyed. Implementations shall define the algorithm, if any, used to recycle a UFiT.

6.3 Addressing Grammar

This section describes the grammar productions needed to form valid SM ME addresses. The production rules and notation are defined in RFC 2234. Implementations shall expose ME addresses that comply with the `MEAddress` production specified by the SM ME addressing grammar as specified in this section. Implementations shall use the UFcT production to address a CIM class. RFC 2234 defines a modified version of Backus-Naur Form (BNF), called Augmented BNF (ABNF), which has proven popular among many Internet specifications. It balances compactness and simplicity with reasonable representational power. RFC 2234 defines a formal set of rules for a formal unambiguous machine-parsable grammar.

The SM ME addressing grammar provides unambiguous productions that guarantee that a programmatic parser can be created to differentiate valid ME addresses from invalid ME addresses. The SM ME addressing rules provide directives to implementations to guarantee that a UFiT is unique within its immediate container.

The following is the SM ME addressing grammar:

```
;;
;; The SM ME Addressing grammar provides a scheme for validating that
;; an address presented to a parser is valid according to the rules
;; specified in the text of this document.
;;
;; In the grammar below, the valid Addressing Associations have been
;; used to formulate valid address patterns by the grammar productions.
;; The grammar assumes that the implementation has used the
```

```
;; valid Addressing Associations named in tables above to form the
;; relationships between the instances represented by the UFiT terms in
;; the address.
;;
;; For example, the address /system1/logs1/log2/record1 is a valid
;; address. The grammar specifies that a UFiT term using the UFcT
;; "system" may be followed by a UFiT term using the UFcT "logs", a term
;; using "logs" may be followed by a term using "log", and a term using
;; "log" may be followed by a term using "record". It is up to the
;; implementation to validate that the correct corresponding CIM
;; associations were used in the CIM server to associate the terms in the
;; address.

;; Managed Element Address
MEAddress = UFiP                                ; objects
MEAddress =/ (UFiP "=>" TargetAssoc [ "=>" UFiP ])  ; associations

;; User Friendly instance Paths:
;; /admin1
;; /admin1/<logical paths>
;; /admin1/<physical paths>
;; /system1
;; /hdwr1
UFiP = TermSeparator AdminDomainUFcT UFiS
UFiP =/ UFiP-Logical
UFiP =/ UFiP-Physical

;; User Friendly instance Tags:
UFiS = %x31-39 *[%x30-39]  ; User Friendly instance Suffix
UFiT = (UFcT UFiS)

;; User Friendly class Tags:
;; All the UFcTs in the Tables plus OEM.
UFcT = TableOneUFcT / TableTwoUFcT / TableThreeUFcT / TableFourUFcT
UFcT =/ TableFiveUFcT / TableSixUFcT / TableSevenUFcT / TableEightUFcT
UFcT =/ TableNineUFcT / TableTenUFcT / TableElevenUFcT
UFcT =/ OEMUFcT

TermSeparator = "\" / "/"

;; User-Friendly instance Paths for Logical Containment Hierarchy
;; (Example paths included prior to productions)
;; /admin1/system1/<system components or collections>
;; /admin1/system1/system1/<system components or collections>
;; /admin1/system1/system1/system1/<system components or collections>
UFiP-Logical = [ TermSeparator AdminDomainUFcT UFiS ]
            1*( TermSeparator ComputerSystemUFcT UFiS )
              [ TermSeparator ( ComponentOfSystem / SystemCollection ) ]

;; ComponentOfSystem is a production for immediately-contained elements
;; of a ComputerSystem
ComponentOfSystem =  Service / SAP / StoragePool / Device
ComponentOfSystem =/ OperatingSystem
```

```
719
720   ;; All ComputerSystem Devices except Sensors
721   ;; /admin1/system1/disk3
722   Device          =  LogicalDeviceUFcT UFiS
723   Device          =/ SensorUFcT UFiS
724   Device          =/ EnetStack
725
726   ;; Operating System
727   ;; /admin1/system1/os1
728   OperatingSystem = OperatingSystemUFcT UFiS
729
730   ;; Service Access Points
731   ;; /admin1/system1/remotesap1
732   SAP             =  ProtoEndptUFcT UFiS
733   SAP             =/ RemoteSAPUFcT UFiS
734   SAP             =/ RemotePortUFcT UFiS
735   SAP             =/ SCSIProtoEndptUFcT UFiS
736   SAP             =/ ServiceAccessURIUFcT UFiS
737   SAP             =/ TextRedirectSAPUFcT UFiS
738   SAP             =/ IPStack
739
740   ;; System services
741   ;; /admin1/system1/bootsvc1
742   Service         =  BootSvcUFcT UFiS
743   Service         =/ CLPSvcUFcT UFiS
744   Service         =/ PowerManagementSvcUFcT UFiS
745   Service         =/ SharedDeviceMgmtSvcUFcT UFiS
746   Service         =/ SoftwareInstallationSvcUFcT UFiS
747   Service         =/ SSHSvcUFcT UFiS
748   Service         =/ StorageCfgSvcUFcT UFiS
749   Service         =/ TelnetSvcUFcT UFiS
750   Service         =/ TextRedirectSvcUFcT UFiS
751   Service         =/ TimeSvcUFcT UFiS
752   Service         =/ IpCfgSvcUFcT UFiS
753
754   ;; System storage pool
755   ;; /admin1/system1/storagepool2
756   StoragePool     = StoragePoolUFcT UFiS
757
758   ;;
759   ;; SystemCollection is a production for named collections that
760   ;; are dedicated to a ComputerSystem.
761   ;;
762   SystemCollection =  JobQueue / Logs
763   SystemCollection =/ SoftwareRepo / Sensors / Consoles
764   SystemCollection =/ Group / Privileges
765   SystemCollection =/ Profiles / Capabilities / Capacities
766   SystemCollection =/ Products / Settings
767
768   ;; Dedicated collection for Capabilities
769   ;; /system1/capabilities1
770   ;; /system1/capabilities1/swinstallsvccap1
771   Capabilities    = CapabilitiesUFcT UFiS
```

```
                [ ( TermSeparator MAPCapUFcT UFiS )
                / ( TermSeparator SharingCapUFcT UFiS )
                / ( TermSeparator PowerMgmtCapUFcT UFiS )
                / ( TermSeparator SoftwareInstallationSvcCapUFcT UFiS)
                / ( TermSeparator StorageCapUFcT UFiS )
                / ( TermSeparator StorageCfgCapUFcT UFiS )
                / ( TermSeparator DHCPCapUFcT UFiS ) ]

;; Dedicated collection for Capacities
;; /system1/capacities1
;; /system1/capacities1/configcapacity23
Capacities      = CapacitiesUFcT UFiS
                [ ( TermSeparator ConfigCapacityUFcT UFiS ) ]

;; Dedicated collection for system Consoles
;; /system1/consoles1
;; /system1/consoles1/textredirectsap1
Consoles          = ConsolesUFcT UFiS
                [ ( TermSeparator TextRedirectSAPUFcT UFiS ) ]

;; Ethernet Stack Production
EnetStack = EthernetPortUFcT UFiS TermSeparator IPStack

;; Dedicated collection for Group
;; /admin1/system1/group5
;; /admin1/system1/group5/account2
Group           = GroupUFcT UFiS
                [ ( TermSeparator AccountUFcT UFiS ) ]

;; Job queue
;; /system1/jobq1
;; /system1/jobq1/job323
JobQueue        = JobQueueUFcT UFiS
                [ ( TermSeparator ConcreteJobUFcT UFiS ) ]

;; Dedicated collection for Logs
;; /system1/logs1
;; /system1/logs1/log1
;; /system1/logs1/log1/record45
Logs            = LogsUFcT UFiS
                [ ( TermSeparator RecordLogUFcT UFiS )
                [ ( TermSeparator LogRecordUFcT UFiS ) ] ]

;; Dedicated collection for Privileges
;; /system1/privileges1
;; /system1/privileges1/authorizedpriv2
Privileges      =  PrivilegesUFcT UFiS
              [ (TermSeparator PrivilegeUFcT UFiS ) ]

;; IP Stack and ProtocolEndpoint productions
IPStack         = LANEndPtUFcT UFiS
                [ TermSeparator IPEndPtUFcT UFiS
                [ TermSeparator ( DhcpEndPtUFcT UFiS
```

```
                        [ TermSeparator DhcpServerUFcT UFiS ])
                / ( DnsEndPtUFcT UFiS [TermSeparator DnsServerUFcT UFiS])
                / (GatewayUFcT UFiS)]]

;; Dedicated collection for Profiles
;; /system1/profiles1
;; /system1/profiles1/profile34
;; /system1/profiles1/subprofile32
Profiles        = ProfilesUFcT UFiS
                [ ( TermSeparator ProfileUFcT UFiS )
                / ( TermSeparator SubProfileUFcT UFiS )
                 ]

;; Dedicated collection for Products
;; /system1/products1
;; /system1/products1/product43
Products        = ProductsUFcT UFiS
                [ ( TermSeparator ProductUFcT UFiS ) ]

;; Dedicated collection for Sensors
;; /system1/sensors1
;; /system1/sensors1/tempsensor4
Sensors         = SensorsUFcT UFiS
                [ ( TermSeparator SensorUFcT UFiS ) ]

;; Dedicated collection for all Settings
;; /system1/settings1
;; /system1/settings1/displaysetting32
Settings        = SettingsUFcT UFiS
                [ (SettingData
                / (TermSeparator IPAssignmentSettings)
                / ( TermSeparator BootConfigSetting ) ) ]

SettingData     = ( TermSeparator DisplaySettingUFcT UFiS )
                / ( TermSeparator MAPSettingUFcT UFiS )
                / ( TermSeparator NICSettingUFcT UFiS )
                / ( TermSeparator StorageSettingUFcT UFiS )

;; Dedicated collection for Boot Configuration SettingData
;; /system1/settings1/bootcfgsetting1
;; /system1/settings1/bootcfgsetting1/bootsetting1
;; /system1/settings1/bootcfgsetting1/nicsetting1
;; /system1/settings1/bootcfgsetting1/bootsrcsetting1
;; /system1/settings1/bootcfgsetting1/bootsrcsetting1/bootsetting2
;; /system1/settings1/bootcfgsetting1/bootsrcsetting1/displaysetting1
BootConfigSetting = BootConfigSettingUFcT UFiS
                  [ ( TermSeparator BootSettingDataUFcT UFiS ) /
                    ( SettingData ) /
                    ( TermSeparator BootSourceSetting ) ]
BootSourceSetting = BootSourceSettingUFcT UFiS
                  [ ( TermSeparator BootSettingDataUFcT UFiS ) /
                    ( SettingData ) ]
```

```
;; IP Assignment Settings for IP Settings
IPAssignmentSettings = IPSettingUFcT UFiS
                     [ ( TermSeparator StaticIPSettingUFcT UFiS )
                     / ( TermSeparator DHCPSettingUFcT )
                     / ( TermSeparator DNSGeneralSettingUFcT )
                     / ( TermSeparator DNSSettingUFcT ) ]

;; Dedicated collection for Software
;; /system1/swrepo1
;; /system1/swrepo1/swid23
SoftwareRepo    = SoftwareRepoUFcT UFiS
                [ ( TermSeparator SoftwareIdUFcT UFiS ) ]

;; User-Friendly instance Paths for Physical Containment Hierarchy
;; /admin1/hdwr1
;; /admin1/hdwr1/<physical package paths>
UFiP-Physical = [ TermSeparator AdminDomainUFcT UFiS ]
                TermSeparator HWConcreteCollectionUFcT UFiS
                [ TermSeparator PhysicalPackage ]

;; PhysicalPackage identifies those elements which the Physical
;; Containment Hierarchy allows
;; to be contained within another PhysicalPackage
PhysicalPackage = Rack / Chassis / PhysicalFrame / Package / Card

;; /admin1/hdwr1/rack1
;; /admin1/hdwr1/rack1/chassis1/...
;; /admin1/hdwr1/rack1/frame1/...
;; /admin1/hdwr1/rack1/card1/...
;; /admin1/hdwr1/rack1/pkg1/...
;; /admin1/hdwr1/rack1/<connector or component>
Rack =  RackUFcT UFiS
Rack =/ RackUFcT UFiS TermSeparator PhysicalPackage
Rack =/ RackUFcT UFiS TermSeparator ConnectorSlot
Rack =/ RackUFcT UFiS TermSeparator Component

;; /admin1/hdwr1/chassis1
;; /admin1/hdwr1/chassis1/rack1/...
;; /admin1/hdwr1/chassis1/frame1/...
;; /admin1/hdwr1/chassis1/card1/...
;; /admin1/hdwr1/chassis1/pkg1/...
;; /admin1/hdwr1/chassis1/<connector or component>
Chassis =  ChassisUFcT UFiS
Chassis =/ ChassisUFcT UFiS TermSeparator PhysicalPackage
Chassis =/ ChassisUFcT UFiS TermSeparator ConnectorSlot
Chassis =/ ChassisUFcT UFiS TermSeparator Component

;; /admin1/hdwr1/frame1
;; /admin1/hdwr1/frame1/card1/...
;; /admin1/hdwr1/frame1/pkg1/...
;; /admin1/hdwr1/frame1/<connector or component>
PhysicalFrame =  PhysicalFrameUFcT UFiS
PhysicalFrame =/ PhysicalFrameUFcT UFiS TermSeparator PhysicalPackage
```

```
PhysicalFrame =/ PhysicalFrameUFcT UFiS TermSeparator ConnectorSlot
PhysicalFrame =/ PhysicalFrameUFcT UFiS TermSeparator Component

;; /admin1/hdwr1/pkg1
;; /admin1/hdwr1/pkg1/card1/...
;; /admin1/hdwr1/pkg1/pkg1/...
;; /admin1/hdwr1/pkg1/<connector or component>
Package =  PackageUFcT UFiS
Package =/ PackageUFcT UFiS TermSeparator PhysicalPackage
Package =/ PackageUFcT UFiS TermSeparator ConnectorSlot
Package =/ PackageUFcT UFiS TermSeparator Component

;; /admin1/hdwr1/card1
;; /admin1/hdwr1/card1/card1/...
;; /admin1/hdwr1/card1/<connector or component>
Card =  ( CardUFcT / SystemCardUFcT ) UFiS
Card =/ ( CardUFcT / SystemCardUFcT ) UFiS TermSeparator PhysicalPackage
Card =/ ( CardUFcT / SystemCardUFcT ) UFiS TermSeparator ConnectorSlot
Card =/ ( CardUFcT / SystemCardUFcT ) UFiS TermSeparator Component

;; /admin1/hdwr1/.../connector1
;; /admin1/hdwr1/.../slot1
;; /admin1/hdwr1/.../slot1/card1
;; /admin1/hdwr1/.../connector1/pkg1
ConnectorSlot = Connector / Slot
Connector =  ConnectorUFcT UFiS
Connector =/ ConnectorUFcT UFiS TermSeparator PhysicalPackage
Slot = SlotUFcT UfiS
Slot =/ SlotUFcT UFiS TermSeparator PhysicalPackage

;; /admin1/hdwr1/.../component1
;; /admin1/hdwr1/.../chip1
Component = ( ComponentUFcT / ChipUFcT / PhysicalMemoryUFcT ) UFiS

;; The following sections define the terminal productions for the
;; SM ME Addressing Grammar

;; User Friendly class Tags in the Logical Containment Hierarchy
;; in the order introduced in the UFcT tables in this document.
;; The ordering is based upon the UFcT class' end-most subclass.

;; special opaque OEM specific UFcT
OEMUFcT = "OEM" 1*VCHAR    ; VCHAR: visible characters

;; Table 1 CIM_ManagedElement UFcTs
TableOneUFcT = AuthorizedPrivilegeUFcT / ConfigCapacityUFcT
TableOneUFcT =/ LocationUFcT / LogRecordUFcT / NamespaceUFcT
TableOneUFcT =/ PhysicalLocationUFcT / ProductUFcT / ProfileUFcT
TableOneUFcT =/ SubProfileUFcT

AuthorizedPrivilegeUFcT = "authorizedpriv"
ConfigCapacityUFcT = "configcapacity"
LocationUFcT = "location"
```

```
LogRecordUFcT = "record"
NamespaceUFcT = "namespace"
PhysicalLocationUFcT = "plocation"
ProductUFcT = "product"
ProfileUFcT = "profile"
SubProfileUFcT = "subprofile"

;; Table 2 CIM_Capabilities UFcTs
TableTwoUFcT = MapCapUFcT / PowerMgmtCapUFcT / SharingCapUFcT
TableTwoUFcT =/ SoftwareInstSvcCapUFcT / StorageCapUFcT
TableTwoUFcT =/ StorageCfgCapUFcT
TableTwoUFcT =/ DHCPCapUFcT

MAPCapUFcT = "mapcap"
PowerMgmtCapUFcT = "pwrmgtcap"
SharingCapUFcT = "sharingcap"
SoftwareInstSvcCapUFcT = "swinstallsvccap"
StorageCapUFcT = "storagecap"
StorageCfgCapUFcT = "storagecfgcap"
DHCPCapUFcT = "dhcpcap"

;; Table 3 CIM_SettingData UFcTs
TableThreeUFcT = BootConfigSettingUFcT / BootSourceSettingUFcT
TableThreeUFcT =/ BootSettingDataUFcT / DisplaySettingUFcT
TableThreeUFcT =/ MAPSettingUFcT / NICSettingUFcT / StorageSettingUFcT
TableThreeUFcT =/ IPSettingUFcT / StaticIPSettingUFcT / DHCPSettingUFcT
TableThreeUFcT =/ DNSSettingUFcT / DNSGeneralSettingUFcT

BootConfigSettingUFcT = "bootcfgsetting"
BootSourceSettingUFcT = "bootsrcsetting"
BootSettingDataUFcT = "bootsetting"
DisplaySettingUFcT = "displaysetting"
MAPSettingUFcT = "mapsetting"
NICSettingUFcT = "nicsetting"
StorageSettingUFcT = "storagesetting"
IPSettingUFcT = "ipsettings"
StaticIPSettingUFcT = "staticipsettings"
DHCPSettingUFcT = "dhcpsettings"
DNSSettingUFcT = "dnssettings"
DNSGeneralSettingUFcT = "dnsgeneralsettings"

;; Table 4 CIM_Collection UFcTs
TableFourUFcT = CapabilitiesUFcT / CapacitiesUFcT / ConsolesUFcT
TableFourUFcT =/ ConcreteCollectionUFcT / GroupUFcT / LogsUFcT
TableFourUFcT =/ HWConcreteCollectionUFcT / PrivilegesUFcT
TableFourUFcT =/ ProfilesUFcT / ProductsUFcT / RedundancySetUFcT
TableFourUFcT =/ SettingsUFcT / SensorsUFcT / SoftwareRepoUFcT

;;
;; The UFiP grammar will enforce the UFcT(s) that can be
;; contained in each collection.
;;
CapabilitiesUFcT = "capabilities"
```

```
CapacitiesUFcT = "capacities"
ConsolesUFcT = "consoles"
ConcreteCollectionUFcT = "concretecollection"
GroupUFcT = "group"
LogsUFcT = "logs"
HWConcreteCollectionUFcT = "hdwr"
PrivilegesUFcT = "privileges"
ProfilesUFcT = "profiles"
ProductsUFcT = "products"
RedundancySetUFcT = "redundancyset"
SettingsUFcT = "settings"
SensorsUFcT = "sensors"
SoftwareRepoUFcT = "swrepo"

;; Table 5 CIM_LogicalElement UFcTs
TableFiveUFcT = SoftwareIdUFcT / StoragePoolUFcT

SoftwareIdUFcT = "swid"
StoragePoolUFcT = "storagepool"

;; Table 6 CIM_EnabledLogicalElement UfcTs
TableSixUFcT =  AccountUFcT / ConcreteJobUFcT / JobQueueUFcT
TableSixUFcT =/ OperatingSystemUFcT / RecordLogUFcT

AccountUFcT = "account"
ConcreteJobUFcT = "job"
JobQueueUFcT = "jobq"
OperatingSystemUFcT = "os"
RecordLogUFcT = "log"

;; Table 7 CIM_System UFcTs
TableSevenUFcT = AdminDomainUFcT / ComputerSystemUFcT

AdminDomainUFcT  = "admin"
ComputerSystemUFcT =  "system"
ComputerSystemUFcT =/ "modular"
ComputerSystemUFcT =/ "accessserver" /" blockserver"
ComputerSystemUFcT =/ "bridge" / "chassismgr" / "extender"
ComputerSystemUFcT =/ "fileserver" / "firewall" / "gateway" / "hub"
ComputerSystemUFcT =/ "ioserver" / "ipphone" / "map" / "management"
ComputerSystemUFcT =/ "medialib" / "mobile" / "modular"/ "nas"
ComputerSystemUFcT =/ "nashead" / "printserver" "repeater"
ComputerSystemUFcT =/ "router" / "sp" / “storage" / "storagevlizer"
ComputerSystemUFcT =/ "switch" / "webcache" / "ups"

;; Table 8 CIM_LogicalDevice UFcTs
TableEightUFcT = AlarmUFcT / BatteryUFcT / CDROMDriveUFcT
TableEightUFcT =/ CoolingDeviceUFcT / DAPortUFcT / DiskDriveUFcT
TableEightUFcT =/ DisketteDriveUFcT / DiskPartitionUFcT / DisplayUFcT
TableEightUFcT =/ DVDDriveUFcT / EthernetPortUFcT / FanUFcT / FCPortUFcT
TableEightUFcT =/ HeatPipeUFcT / IBPortUFcT / KeyboardUFcT
TableEightUFcT =/ LogicalDiskUFcT / LogicalModularUFcT / LogicalPortUFcT
TableEightUFcT =/ MediaAccessUFcT / MemoryUFcT / ModemUFcT
```

```
TableEightUFcT =/ NetworkPortUFcT / PCIDeviceUFcT / PCIBridgeUFcT
TableEightUFcT =/ PointingDeviceUFcT / PortControllerUFcT
TableEightUFcT =/ PowerSupplyUFcT / PrinterUFcT / ProcessorUFcT
TableEightUFcT =/ RefrigerationUFcT / SCSIProtoControlUFcT / SensorUFcT
TableEightUFcT =/ SPIPort / StorageVolumeUFcT / StorageExtentUFcT
TableEightUFcT =/ SerialPortUFcT / TapeDriveUFcT / USBPortUFcT
TableEightUFcT =/ WatchdogUFcT / WirelessPortUFcT
TableEightUFcT =/ PortCtrlUFcT

AlarmUFcT = "alarm"
BatteryUFcT = "battery"
CDROMDriveUFcT = "cd"
CoolingDeviceUFcT = "cooling"
DAPortUFcT = "daport"
DiskDriveUFcT = "diskdrive"
DisketteDriveUFcT = "floppy"
DiskPartitionUFcT = "diskpartition"
DisplayUFcT = "display"
DVDDriveUFcT = "dvd"
EthernetPortUFcT = "enetport"
FanUFcT = "fan"
FCPortUFcT = "fcport"
HeatPipeUFcT = "heatpipe"
IBPortUFcT = "ibport"
KeyboardUFcT = "keyboard"
LogicalDiskUFcT = "disk"
LogicalModularUFcT = "logicalmodule" / "blademodule" / "linecard"
LogicalModularUFcT =/ "devicetray"
LogicalPortUFcT = "logicalport"
MediaAccessUFcT = "mediaaccess"
MemoryUFcT = "memory"
ModemUFcT = "modem"
NetworkPortUFcT = "netport"
PCIDeviceUFcT = "pcidev"
PCIBridgeUFcT = "pcibridge"
PointingDeviceUFcT = “pointer” / "mouse" / "trackball"
PointingDeviceUFcT =/ "touchpad" / "touchscreen"
PortControllerUFcT = "portctlr" / "nic" / "hca" / "tca" / "hba"
PowerSupplyUFcT = "pwrsupply"
PrinterUFcT = "printer"
ProcessorUFcT = "cpu"
RefrigerationUFcT = "refrigeration"
SCSIProtoControlUFcT = "scsiprotctlr"
SensorUFcT =  "sensor" / "currentsensor" / "tachsensor"
SensorUFcT =/ "tempsensor" / "voltsensor" / "humiditysensor"
SensorUFcT =/ "countersensor" / "swsensor" / "locksensor"
SensorUFcT =/ "smokesensor"/ "airsensor" / "presencesensor"
SensorUFcT =/ "numsensor" / "ncurrentsensor" / "ntachsensor"
SensorUFcT =/ "ntempsensor" / "nvoltsensor" / "nhumiditysensor"
SensorUFcT =/ "ncountersensor" / "nswsensor" / "nlocksensor"
SensorUFcT =/ "nsmokesensor"/ "nairsensor" / "npresencesensor"
SPIPort = "spiport"
StorageVolumeUFcT = "storagevol"
```

```
StorageExtentUFcT = "storageext"
SerialPortUFcT = "serialport"
TapeDriveUFcT = "tapedrive"
USBPortUFcT = "usbport"
WatchdogUFcT = "watchdog"
WirelessPortUFcT = "wifiport"
PortCtrlUFcT = "portctrl"

;; Convenience Production for all LogicalDevice instances
;; immediately contained in a System
LogicalDeviceUFcT =  AlarmUFcT / BatteryUFcT / CDROMDriveUFcT
LogicalDeviceUFcT =/ CoolingDeviceUFcT / DAPortUFcT / DiskDriveUFcT
LogicalDeviceUFcT =/ DisketteDriveUFcT / DiskPartitionUFcT / DisplayUFcT
LogicalDeviceUFcT =/ DVDDriveUFcT / EthernetPortUFcT / FanUFcT
LogicalDeviceUFcT =/ FCPortUFcT / HeatPipeUFcT / IBPortUFcT
LogicalDeviceUFcT =/ KeyboardUFcT  / LogicalDiskUFcT /
LogicalDeviceUFcT =/ LogicalModularUFcT / LogicalPortUFcT
LogicalDeviceUFcT =/ MediaAccessUFcT / MemoryUFcT / ModemUFcT
LogicalDeviceUFcT =/ NetworkPortUFcT / PCIDeviceUFcT / PCIBridgeUFcT
LogicalDeviceUFcT =/ PointingDeviceUFcT / PortControllerUFcT
LogicalDeviceUFcT =/ PowerSupplyUFcT / PrinterUFcT / ProcessorUFcT
LogicalDeviceUFcT =/ RefrigerationUFcT / SCSIProtoControlUFcT
LogicalDeviceUFcT =/ SPIPort / StorageVolumeUFcT / StorageExtentUFcT
LogicalDeviceUFcT =/ SerialPortUFcT / TapeDriveUFcT / USBPortUFcT
LogicalDeviceUFcT =/ WatchdogUFcT / WirelessPortUFcT
LogicalDeviceUFcT =/ PortCtrlUFcT

;; Table 9
TableNineUFcT =  BootSvcUFcT / CLPSvcUFcT / PowerManagementSvcUFcT
TableNineUFcT =/ SharedDeviceMgmtSvcUFcT / SoftwareInstallationSvc
TableNineUFcT =/ SSHSvcUFcT / StorageCfgSvcUFcT / TelnetSvcUFcT
TableNineUFcT =/ TextRedirectSvcUFcT / TimeSvcUFcT
TableNineUFcT =/ IpCfgSvcUFcT

BootSvcUFcT = "bootsvc"
CLPSvcUFcT = "clpsvc"
PowerManagementSvcUFcT = "pwrmgtsvc"
SharedDeviceMgmtSvcUFcT = "shareddevicesvc"
SoftwareInstallationSvc = "swinstallsvc"
SSHSvcUFcT = "sshsvc"
StorageCfgSvcUFcT = "storagecfgsvc"
TelnetSvcUFcT = "telnetsvc"
TextRedirectSvcUFcT = "textredirectsvc"
TimeSvcUFcT = "timesvc"
IPCfgSvcUFcT = "ipcfgsvc"

;; Table 10
TableTenUFcT = ProtoEndptUFcT / RemoteSAPUFcT / RemotePortUFcT
TableTenUFcT =/ SCSIProtoEndptUFcT / ServiceAccessURIUFcT
TableTenUFcT =/ TextRedirectSAPUFcT
TableTenUFcT =/ IPEndPtUFcT / DNSEndPtUFcT / LANEndPtUFcT
TableTenUFcT =/ DhcpEndPtUFcT / DnsServerUFcT / DhcpServerUFcT
TableTenUFcT =/ GatewayUFcT
```

```
ProtoEndptUFcT = "protoendpt"
RemoteSAPUFcT = "remotesap"
RemotePortUFcT = "remoteport"
SCSIProtoEndptUFcT = "scsiendpt"
ServiceAccessURIUFcT = "serviceuri"
TextRedirectSAPUFcT = "textredirectsap"
IPEndPtUFcT = "ipendpt"
DNSEndPtUFcT = "dnsendpt"
LANEndPtUFcT = "lanendpt"
DhcpEndPtUFcT = "dhcpendpt"
DnsServerUFcT = "dnsserver"
DhcpServerUFcT = "dhcpserver"
GatewayUFcT = "gateway"

;; User Friendly class Tags in the Physical Containment Hierarchy:

;; Table 11 CIM_PhysicalElement UFcTs
TableElevenUFcT = CardUFcT / ChassisUFcT / ChipUFcT / ComponentUFcT
TableElevenUFcT =/ ConnectorUFcT / FrameUFcT / PackageUFcT
TableElevenUFcT =/ PhysicalMemoryUFcT / RackUFcT SlotUFcT
TableElevenUFcT =/ SystemCardUFcT

CardUFcT = "card"

ChassisUFcT =  "chassis"
ChassisUFcT =/ "laptop" / "desktop" / "tower"
ChassisUFcT =/ "storagechas" / "notebook" / "mainchassis"
ChassisUFcT =/ "expansion" / "peripheralchassis" / "subchassis"

ChipUFcT =  "bga" / "chip" / "dimm" / "dip" / "fpbga" / "lc"
ChipUFcT =/ "lga" / "plcc" / "pga"  / “propchip" / "qfp" / "rimm"
ChipUFcT =/ "sip" / "simm"  / "smd" / "sodimm" / "soj" / "ssmp"
ChipUFcT =/ "soic" / "srimm" / "tqfp" / "tsop" / "zip"

ComponentUFcT = "component"

ConnectorUFcT = "connector"

PhysicalFrameUFcT = "frame"

;; generic package UFcT
PackageUFcT =  "pkg"
PackageUFcT =/ "backplanepkg" / "batterypkg" / "bladepkg" / "bladexpkg"
PackageUFcT =/ "cardpkg” / "chassispkg" / "cpupkg"/ "diskpkg"
PackageUFcT =/ "fanpkg" / "framepkg" /"memorypkg" / "modulepkg"
PackageUFcT =/ "pwrpkg" / "pwrsrcpkg" / "rackpkg" / "sensorpkg"
PackageUFcT =/ "storagepkg"

PhysicalMemoryUFcT = "bram" / "cache" / "cdram" / "ddr" / "dram" / "edo"
PhysicalMemoryUFcT =/ "edram" / "eeprom" / "eprom" / "flash" / "pmem"
PhysicalMemoryUFcT =/ "ram" / "rdram" / "sdram" / "sgram" / "sram"
PhysicalMemoryUFcT =/ "synchdram" / "vram"
```

```
RackUFcT = "rack"

SlotUFcT = "slot"

SystemCardUFcT =  "accesscard" /"agpcard" / "buscard" / "eisacard"
SystemCardUFcT =/ "giocard" / "hiocard" / "ibcard"
SystemCardUFcT =/ "isacard" / "mcacard" / "nubuscard"
SystemCardUFcT =/ "pccard" /"pcicard" / "pciecard" / "pcixcard"
SystemCardUFcT =/ "pcmciacard" / "pmccard" / "sbuscard"
SystemCardUFcT =/ "vesacard" / "vmecard" / "xiocard"

;;
;; Associations as the address targets:
TargetAssoc =  "SystemComponent" / "SystemDevice"
TargetAssoc =/ "HostedService" / "HostedAccessPoint"
TargetAssoc =/ "OwningCollectionElement" / "HostedPool"
TargetAssoc =/ "ConcreteComponent" / "MemberOfCollection"
TargetAssoc =/ "OrderedMemberOfCollection" / "LogManagesRecord"
TargetAssoc =/ "HostedJobQueue" / "JobDestinationJobs"
TargetAssoc =/ "MemberOfCollection"
TargetAssoc =/ "ChassisInRack" / "PackageInChassis"
TargetAssoc =/ "Container" / "PackagedComponent"
TargetAssoc =/ "ConnectorOnPackage" / "PackageInConnector"
TargetAssoc =/ "PackageInSlot" / "CardInSlot" / "CardOnCard"
TargetAssoc =/ "AssociatedAlarm" / "BootDevice" / "ScopedSetting"
TargetAssoc =/ "AssociatedCooling" /"ElementSoftwareIdentity"
TargetAssoc =/ "ServiceAvailableToElement"
TargetAssoc =/ "SAPAvailableForElement"
TargetAssoc =/ "DeviceSAPImplementation"
TargetAssoc =/ "DiskPartitionBasedOnVolume"
TargetAssoc =/ "ElementCapabilities"
TargetAssoc =/ "MediaPresent" / "SAPAvailableToElement"
TargetAssoc =/ "AllocatedFromStoragePool" / "Realizes"
TargetAssoc =/ "UseOfMessageLog" / "OperationLog"
TargetAssoc =/ "ControlledBy" / "ComputerSystemPackage"
TargetAssoc =/ "InstalledSoftwareIdentity" / "HostedDependency"
TargetAssoc =/ "RunningOS" / "InstalledOS"
TargetAssoc =/ "AssociatedBattery"
TargetAssoc =/ "SuppliesPower" / "AssociatedSensor"
TargetAssoc =/ "SCSIInitiatorTargetLogicalUnitPath"/ "BindsTo"
TargetAssoc =/ "ActiveConnection" / "PortImplementsEndpoint"
TargetAssoc =/ "ProvidesEndpoint" / "SASSAPDependency"
TargetAssoc =/ "ServiceServiceDependency" / "AuthorizedTarget"
TargetAssoc =/ "AuthorizedSubject" / "AccountOnSystem"
TargetAssoc =/ "ProtocolControllerForUnit"
TargetAssoc =/ "AssociatedPowerManagementService"
TargetAssoc =/ "SharingDependency" / "ElementCapacity"
TargetAssoc =/ "ServiceAffectsElement"
TargetAssoc =/ "BindsTo" / "PortImplementsEndpoint"
TargetAssoc =/ "RemoteAccessAvailableToElement" / "EndpointIdentity" TargetAssoc =/
"OrderedComponent" / "ProtocolControllerForPort"
TargetAssoc =/ "ServiceAvailableToElement"
```

ANNEX A
(informative)

Considerations for Implementation

Table A-1 shows several examples of valid SM UFiTs that are based on the instance's UFcT and are unique within the container. The first column shows the container instance's UFcT followed by the corresponding CIM class name in parentheses. Similarly, the second column shows the contained instance's UFcT with its corresponding CIM class name in parentheses. The third column shows the contained instance's UFiT followed by an example of a complete UFiP. Note that the difference between the UFcT and the UFiT is the instance suffix (a positive non-zero integer). The instance suffix must be unique within the scope of the container class.

Table A-1 – Examples of SM UFiTs

Container UFcT (CIM Class Name)	**Contained UFcT** (CIM Class Name)	**Contained Instance UFiT** (Complete UFiP)
admin (CIM_AdminDomain)	hdwr (CIM_ConcreteCollection)	hdwr1 /admin1/hdwr1
hdwr (CIM_ConcreteCollection)	rack (CIM_Rack)	rack1 /admin1/hdwr1/rack1
rack (CIM_Rack)	chassis (CIM_Chassis)	Chassis4 /admin1/hdwr1/rack2/chassis4 /admin1/hdwr1/rack11/chassis4
rack (CIM_Rack)	iochassis (CIM_Chassis)	iochassis4 /admin1/hdwr1/rack2/iochassis4 /admin1/hdwr1/rack11/iochassis4
chassis (CIM_Chassis)	pkg (CIM_PhysicalPackage)	Pkg2 /admin1/hdwr1/chassi4/pkg2 /admin1/hdwr1/rack1/chassis1/pkg2
chassis (CIM_Chassis)	bladepkg (CIM_PhysicalPackage)	bladepkg2 /admin1/hdwr1/chassi4/bladepkg2 /admin1/hdwr1/rack1/chassis1/bladepkg2
pkg (CIM_PhysicalPackage)	diskpkg (CIM_PhysicalPackage)	Diskpkg1 /admin1/hdwr1/pkg1/diskpkg1 /admin1/hdwr1/rack1/pkg2/diskpkg1 /admin1/hdwr1/rack1/chassis2/pkg5/diskpkg1 /admin1/hdwr1/chassis4/pkg3/diskpkg1
bladepkg (CIM_PhysicalPackage)	pwrpkg (CIM_PhysicalPackage)	pwrpkg2 pwrpkg3 /admin1/hdwr1/bladepkg2/pwrpkg2 /admin1/hdwr1/bladepkg5/pwrpkg2 /admin1/hdwr1/rack7/bladechassis2/bladepkg8/pwrpkg3
pkg (CIM_PhysicalPackage)	chip (CIM_Chip)	chip1 chip7 /admin1/hdwr1/pkg1/chip1 /admin1/hdwr1/pkg1/chip7 /admin1/hdwr1/chassis2/pkg1/chip1
bladepkg (CIM_PhysicalPackage)	chip (CIM_Chip)	chip1 /admin1/hdwr1/bladepkg2/chip1 /admin1/hdwr1/chassis4/bladepkg4/chip1

Container UFcT (CIM Class Name)	**Contained UFcT** (CIM Class Name)	**Contained Instance UFiT** (Complete UFiP)
chassis (CIM_Chassis)	chip (CIM_Chip)	chip1 /admin1/hdwr1/chassis4/chip1
card (CIM_Card)	chip (CIM_Chip)	chip1 /admin1/hdwr1/card1/chip1 /admin1/hdwr1/chassis3/card2/chip1 /admin1/hdwr1/rack18/chassis2/bladepkg8/card2/chip1
bladepkg (CIM_PhysicalPackage)	slot (CIM_Slot)	slot1 slot8 /admin1/hdwr1/bladepkg1/slot1
card (CIM_Card)	slot (CIM_Slot)	slot1 slot4 /admin1/hdwr1/bladepkg3/card1/slot1 /admin1/hdwr1/bladepkg3/card2/slot1 /admin1/hdwr1/bladepkg3/card2/slot4
chassis (CIM_Chassis)	slot (CIM_Slot)	slot1 /admin1/hdwr1/chassis2/slot1 /admin1/hdwr1/rack11/chassis2/slot1
admin (CIM_AdminDomain)	system (CIM_ComputerSystem)	system1, system2, system7, system3, router1, map1 /admin1/system1 /admin1/ map1
system (CIM_ComputerSystem)	system (CIM_ComputerSystem)	system1, map1 /admin1/system1/system1 /admin1/system1/ map1
system (CIM_ComputerSystem)	disk (CIM_LogicalDevice)	disk1 /admin1/system2/disk1 /admin1/system;2/system1/disk1
system (CIM_ComputerSystem)	service (CIM_Service)	service1, service2, service5 /admin1/system3/service5 /admin1/system1/ map1/service5
system (CIM_ComputerSystem)	remotesap (CIM_RemoteServiceAccess Point)	remotesap1, remotesap2, remotesap3, remotesap4 /admin1/system3/ remotesap4 /admin1/system1/system1/ remotesap3
system (CIM_ComputerSystem)	group (CIM_Group)	group1 /admin1/system1/group1
system (CIM_ComputerSystem)	storagepool (CIM_StoragePool)	storagepool1 /admin1/system1/storagepool1
capabilities logs settings products (CIM_ConcreteCollection)	mapcap log swid (CIM_ManagedElement)	capabilities1, logs1, settings1, products1 /admin1/system1/capabilities1/mapcap1 /admin1/system1/ logs1/log1 /admin1/system1/settings1/mapsetting1 /admin1/system1/products1/product1

1310

ANNEX B
(informative)

Document Conventions

B.1 CIM_ Prefix

When referring to CIM classes, this document uses the CIM_ prefix only when necessary for clarity. For example, ComputerSystem, rather than CIM_ComputerSystem, is used unless the context demands the prefix for clarity.

B.2 Notation

Augmented Backus-Naur Form (ABNF) is used in this document to describe various aspects of the SM addressing specification. The complete grammar can be found in 6.3.

The ← notation denotes the direction of containment. The container is on the left of the arrow head and is read as “contains”.

The following fonts are used to indicate specification elements:

- Text in `courier new font` indicates productions and literal characters used in a grammar syntax expression.
- Text in <*italicized*> font indicates a CIM association class.

ANNEX C
(informative)

Change Log

Version	Date	Author	Description
1.0.0a	11/02/2006	A. Merkin	Preliminary Standard
1.0.0	04/23/2009		DMTF Standard Release

Bibliography

DMTF DSP0217, *SMASH Implementation Requirements*, 1.0.0,
http://www.dmtf.org/standards/published_documents/DSP0217_1.0.0.pdf

DMTF DSP2001, *Systems Management Architecture for Server Hardware (SMASH) Command Line Protocol (CLP) Architecture White Paper*, 1.0.0,
http://www.dmtf.org/standards/published_documents/DSP2001_1.0.1.pdf