



1

2

3

4

Document Number: DSP1042

Date: 2007-08-27

Version: 1.0.0a

5 **System Virtualization Profile**

6 **Document Type: Specification**

7 **Document Status: Preliminary Standard**

8 **Document Language: E**

9 [Copyright Notice](#)

10 [Copyright © 2007 Distributed Management Task Force, Inc. \(DMTF\). All rights reserved.](#)

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

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25 claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn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

CONTENTS

29	1	Scope	10
30	2	Normative References.....	10
31	2.1	Approved References	10
32	2.2	References under Development.....	10
33	2.3	Other References.....	11
34	3	Terms and Definitions.....	11
35	4	Symbols and Abbreviated Terms.....	12
36	5	Synopsis.....	13
37	6	Description (Informative)	14
38	6.1	DMTF Management Profile Relationships	14
39	6.2	System Virtualization Class Schema	16
40	6.3	Virtual System Configurations.....	18
41	6.4	Resource Allocation	19
42	6.5	Snapshots	20
43	7	Implementation.....	20
44	7.1	Host System.....	20
45	7.2	Profile Registration.....	20
46	7.2.1	System Virtualization Profile	21
47	7.2.2	Scoped Resource Allocation Profiles (Conditional)	21
48	7.3	Representation of Hosted Virtual Systems	21
49	7.3.1	Profile Conformance for Hosted Virtual Systems	22
50	7.3.2	CIM_VirtualSystemSettingData.VirtualSystemType Property	22
51	7.4	Virtual System Management Capabilities	22
52	7.4.1	CIM_VirtualSystemManagementCapabilities Class	22
53	7.4.2	CIM_VirtualSystemManagementCapabilities.VirtualSystemTypesSupported[]	
54		Array Property (Optional).....	22
55	7.4.3	CIM_VirtualSystemManagementCapabilities.SynchronousMethodsSupported[]	
56		Array Property (Optional).....	22
57	7.4.4	CIM_VirtualSystemManagementCapabilities.AsynchronousMethodsSupported[]	
58		Array Property (Optional).....	22
59	7.4.5	CIM_VirtualSystemManagementCapabilities.IndicationsSupported[] Array	
60		Property (Optional).....	23
61	7.4.6	Grouping Rules for Implementations of Methods of the	
62		CIM_VirtualSystemManagementService Class	23
63	7.5	Virtual System Definition and Modification	24
64	7.5.1	CIM_VirtualSystemSettingData.InstanceID Property	24
65	7.5.2	CIM_VirtualSystemSettingData.ElementName Property (Optional).....	24
66	7.5.3	CIM_VirtualSystemSettingData.VirtualSystemIdentifier Property (Optional).....	24
67	7.5.4	CIM_VirtualSystemSettingData.VirtualSystemType Property (Optional).....	25
68	7.6	Virtual Resource Definition and Modification	25
69	7.7	Virtual System Snapshots (Optional)	25
70	7.7.1	Virtual System Snapshot Service and Capabilities.....	25
71	7.7.2	Virtual System Snapshot Representation (Conditional)	26
72	7.7.3	Designation of the Last Applied Snapshot (Conditional)	26
73	7.7.4	Designation of the Most Current Snapshot in Branch (Conditional).....	26
74	7.7.5	Virtual System Snapshot Capabilities (Optional)	27
75	8	Methods.....	27
76	8.1	General Behavior of Extrinsic Methods	27
77	8.1.1	Resource Allocation Requests.....	27
78	8.1.2	Method Results	28
79	8.1.3	Asynchronous Processing	28
80	8.2	Methods of the VirtualSystemManagementService Class.....	29
81	8.2.1	CIM_VirtualSystemManagementService.DefineSystem() Method (Conditional)	29
82	8.2.2	CIM_VirtualSystemManagementService.DestroySystem() Method (Conditional)	31

83	8.2.3	CIM_VirtualSystemManagementService.AddResourceSettings() Method (Conditional).....	32
84			
85	8.2.4	CIM_VirtualSystemManagementService.ModifyResourceSettings() Method (Conditional).....	33
86			
87	8.2.5	CIM_VirtualSystemManagementService.ModifySystemSettings() Method (Conditional).....	35
88			
89	8.2.6	CIM_VirtualSystemManagementService.RemoveResourceSettings() Method (Conditional).....	36
90			
91	8.3	Methods of the CIM_VirtualSystemSnapshotService Class.....	37
92	8.3.1	CIM_VirtualSystemSnapshotService.CreateSnapshot() Method (Conditional).....	37
93	8.3.2	VirtualSystemSnapshotService.DestroySnapshot() Method (Conditional).....	39
94	8.3.3	VirtualSystemSnapshotService.ApplySnapshot() Method (Conditional).....	40
95	8.4	Profile Conventions for Operations.....	41
96	8.4.1	CIM_AffectedJobElement.....	41
97	8.4.2	CIM_ComputerSystem.....	42
98	8.4.3	CIM_ConcreteJob.....	42
99	8.4.4	CIM_Dependency.....	42
100	8.4.5	CIM_ElementCapabilities.....	42
101	8.4.6	CIM_ElementConformsToProfile.....	42
102	8.4.7	CIM_HostedDependency.....	43
103	8.4.8	CIM_HostedService.....	43
104	8.4.9	CIM_LastAppliedSnapshot.....	43
105	8.4.10	CIM_MostCurrentSnapshotInBranch.....	44
106	8.4.11	CIM_ReferencedProfile.....	44
107	8.4.12	CIM_RegisteredProfile.....	44
108	8.4.13	CIM_ServiceAffectsElement.....	45
109	8.4.14	CIM_SnapshotOfVirtualSystem.....	45
110	8.4.15	CIM_System.....	45
111	8.4.16	CIM_VirtualSystemManagementCapabilities.....	45
112	8.4.17	CIM_VirtualSystemManagementService.....	45
113	8.4.18	CIM_VirtualSystemSnapshotService.....	45
114	8.4.19	CIM_VirtualSystemSnapshotCapabilities.....	45
115	8.4.20	CIM_VirtualSystemSnapshotServiceCapabilities.....	46
116	9	Use Cases (Informative).....	46
117	9.1	General Assumptions.....	46
118	9.2	Discovery, Localization, and Inspection.....	46
119	9.2.1	SLP-Based Discovery of CIM Object Managers Hosting Implementations of This Profile.....	48
120			
121	9.2.2	Locate Conformant Implementations Using the EnumerateInstances() Operation.....	48
122			
123	9.2.3	Locate Conformant Implementations Using the ExecuteQuery() Operation.....	48
124	9.2.4	Locate Host Systems That Are Central and Scoping Instances of This Profile.....	49
125	9.2.5	Locate Implementations of Scoped Resource Allocation DMTF Management Profiles.....	49
126			
127	9.2.6	Locate Virtual System Management Service.....	50
128	9.2.7	Determine the Capabilities of an Implementation.....	50
129	9.2.8	Locate Hosted Resource Pools of a Particular Resource Type.....	51
130	9.2.9	Obtain a Set of Central Instances of Scoped Resource Allocation DMTF Management Profiles.....	51
131			
132	9.2.10	Determine the Supported Resource Types of an Implementation.....	52
133	9.2.11	Determine the Default Resource Pool for a Resource Type.....	53
134	9.2.12	Determine Resource Pool for a Resource Allocation Request or Allocated Resource.....	54
135			
136	9.2.13	Determine Valid Settings for a Resource Type.....	54
137	9.2.14	Determine Implementation Class Specifics.....	56
138	9.2.15	Determine the Implementation Class for a Resource Type.....	56
139	9.2.16	Locate Virtual Systems Hosted by a Host System.....	57

140 9.3 Virtual System Definition, Modification, and Destruction 57

141 9.3.1 Virtual System Definition 57

142 9.3.2 Virtual System Modification 59

143 9.3.3 Destroy Virtual System 63

144 9.4 Snapshot-Related Activities 63

145 9.4.1 Locate Virtual System Snapshot Service 66

146 9.4.2 Determine Capabilities of a Virtual System Snapshot Service 66

147 9.4.3 Create Snapshot 67

148 9.4.4 Locate Snapshots of a Virtual System 67

149 9.4.5 Locate the Source Virtual System of a Snapshot 67

150 9.4.6 Locate the Most Current Snapshot in a Branch of Snapshots 68

151 9.4.7 Locate Dependent Snapshots 68

152 9.4.8 Locate Parent Snapshot 69

153 9.4.9 Apply Snapshot 69

154 9.4.10 Destroy Snapshot 70

155 10 CIM Elements 70

156 10.1 CIM_AffectedJobElement (Conditional) 71

157 10.2 CIM_ConcreteJob (Conditional) 71

158 10.3 CIM_Dependency (Conditional) 72

159 10.4 CIM_ElementCapabilities (Host System) 72

160 10.5 CIM_ElementCapabilities (Virtual System Management Service) (Conditional) 72

161 10.6 CIM_ElementCapabilities (Virtual System Snapshot Service) (Conditional) 73

162 10.7 CIM_ElementCapabilities (Snapshots of Virtual Systems) (Conditional) 73

163 10.8 CIM_ElementConformsToProfile 74

164 10.9 CIM_HostedDependency 74

165 10.10 CIM_HostedService (Virtual System Management Service) (Conditional) 75

166 10.11 CIM_HostedService (Virtual System Snapshot Service) (Conditional) 75

167 10.12 CIM_LastAppliedSnapshot (Conditional) 76

168 10.13 CIM_MostCurrentSnapshotInBranch (Conditional) 76

169 10.14 CIM_ReferencedProfile (Conditional) 77

170 10.15 CIM_RegisteredProfile 77

171 10.16 CIM_ServiceAffectsElement (Virtual System Management Service) (Conditional) 77

172 10.17 CIM_ServiceAffectsElement (Virtual System Snapshot Service) (Conditional) 78

173 10.18 CIM_SnapshotOfVirtualSystem (Conditional) 78

174 10.19 CIM_System 79

175 10.20 CIM_VirtualSystemManagementCapabilities 79

176 10.21 CIM_VirtualSystemManagementService (Conditional) 79

177 10.22 CIM_VirtualSystemSettingData (Input) (Conditional) 80

178 10.23 CIM_VirtualSystemSettingData (Snapshot) (Conditional) 81

179 10.24 CIM_VirtualSystemSnapshotCapabilities (Optional) 81

180 10.25 CIM_VirtualSystemSnapshotService (Optional) 82

181 10.26 CIM_VirtualSystemSnapshotServiceCapabilities (Conditional) 82

182

183 **Figures**

184 Figure 1 – DMTF Management Profiles Related to System Virtualization 15

185 Figure 2 – *System Virtualization Profile*: Class Diagram 17

186 Figure 3 – System Virtualization Profile Instance Diagram: Discovery, Localization, and Inspection 47

187 Figure 4 – Virtual System Configuration Based on "Input" Virtual System Configurations and
188 Implementation Defaults 58

189 Figure 5 – Virtual System Resource Modification 62

190 Figure 6 – System Virtualization Profile: Snapshot Example 66

191

192 **Tables**

193	Table 1 – Related Profiles	13
194	Table 2 – DefineSystem() Method: Parameters	29
195	Table 3 – DefineSystem() Method: Return Code Values	31
196	Table 4 – DestroySystem() Method: Parameters	32
197	Table 5 – DestroySystem() Method: Return Code Values	32
198	Table 6 – AddResourceSettings() Method: Parameters.....	32
199	Table 7 – AddResourceSettings() Method: Return Code Values	33
200	Table 8 – ModifyResourceSettings() Method: Parameters	34
201	Table 9 – ModifyResourceSettings() Method: Return Code Values.....	35
202	Table 10 – ModifySystemSettings() Method: Parameters.....	35
203	Table 11 – ModifySystemSettings() Method: Return Code Values	36
204	Table 12 – RemoveResourceSettings() Method: Parameters	36
205	Table 13 – RemoveResourceSettings() Method: Return Code Values.....	37
206	Table 14 – CreateSnapshot() Method: Parameters	37
207	Table 15 – CreateSnapshot() Method: Return Code Values.....	38
208	Table 16 – DestroySnapshot() Method: Parameters.....	39
209	Table 17 – DestroySnapshot() Method: Return Code Values	40
210	Table 18 – ApplySnapshot() Method: Parameters	40
211	Table 19 – ApplySnapshot() Method: Return Code Values.....	40
212	Table 20 – Operations: CIM_AffectedJobElement Association.....	41
213	Table 21 – Operations: CIM_Dependency Association.....	42
214	Table 22 – Operations: CIM_ElementCapabilities Association	42
215	Table 23 – Operations: CIM_ElementConformsToProfile Association	43
216	Table 24 – Operations: CIM_HostedDependency Association	43
217	Table 25 – Operations: CIM_HostedService Association	43
218	Table 26 – Operations: CIM_LastAppliedSnapshot Association	44
219	Table 27 – Operations: CIM_MostCurrentSnapshotInBranch Association	44
220	Table 28 – Operations: CIM_ReferencedProfile Association	44
221	Table 29 – Operations: CIM_ServiceAffectsElement Association.....	45
222	Table 30 – Operations: CIM_SnapshotOfVirtualSystem Association.....	45
223	Table 31 – CIM Elements: System Virtualization Profile.....	70
224	Table 32 – Association: CIM_AffectedJobElement	71
225	Table 33 – Class: CIM_ConcreteJob	71
226	Table 34 – Class: CIM_Dependency Class.....	72
227	Table 35 – Association: CIM_ElementCapabilities (Host System).....	72
228	Table 36 – Association: CIM_ElementCapabilities (Virtual System Management).....	73
229	Table 37 – Association: CIM_ElementCapabilities (Snapshot Service).....	73
230	Table 38 – Association: CIM_ElementCapabilities (Snapshots of Virtual Systems)	74
231	Table 39 – Association: CIM_ElementConformsToProfile.....	74
232	Table 40 – Association: CIM_HostedDependency.....	74
233	Table 41 – Association: CIM_HostedService (Virtual System Management Service)	75
234	Table 42 – Association: CIM_HostedService (Virtual System Snapshot Service)	75
235	Table 43 – Association: CIM_LastAppliedSnapshot.....	76
236	Table 44 – Association: CIM_MostCurrentSnapshotInBranch	76
237	Table 45 – Association: CIM_ReferencedProfile.....	77
238	Table 46 – Class: CIM_RegisteredProfile	77

239 Table 47 – Association: CIM_ServiceAffectsElement (Virtual System Management Service) 78

240 Table 48 – Association: CIM_ServiceAffectsElement 78

241 Table 49 – Association: CIM_SnapshotOfVirtualSystem 79

242 Table 50 – Class: CIM_VirtualSystemManagementCapabilities 79

243 Table 51 – Class: CIM_VirtualSystemManagementCapabilities 79

244 Table 52 – Class: CIM_VirtualSystemManagementService 80

245 Table 53 – Class: CIM_VirtualSystemSettingData (Input) 80

246 Table 54 – Class: CIM_VirtualSystemSettingData (Snapshot) 81

247 Table 55 – Class: CIM_VirtualSystemSnapshotCapabilities 82

248 Table 56 – Class: CIM_VirtualSystemSnapshotService 82

249 Table 57 – Class: CIM_VirtualSystemSnapshotServiceCapabilities 82

250

251

Foreword

252 The *System Virtualization Profile* (DSP1042) was prepared by the System Virtualization, Partitioning and
253 Clustering Working Group of the DMTF.

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

256

Introduction

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

263

System Virtualization Profile

264 1 Scope

265 The *System Virtualization Profile* is an autonomous profile that specifies the minimum top-level object
266 model needed for the representation of host systems and the discovery of hosted virtual computer sys-
267 tems. In addition, it specifies a service for the manipulation of virtual computer systems and their
268 resources, including operations for the creation, deletion, and modification of virtual computer systems
269 and operations for the addition or removal of virtual resources to or from virtual computer systems.

270 2 Normative References

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

274 2.1 Approved References

- 275 DMTF [DSP0004](#), *CIM Infrastructure Specification 2.3.0*
276 DMTF [DSP0200](#), *CIM Operations over HTTP 1.2.0*
277 DMTF [DSP0201](#), *Specification for the Representation of CIM in XML 2.2.0*
278 DMTF [DSP1000](#), *Management Profile Specification Template 1.0*
279 DMTF [DSP1001](#), *Management Profile Specification Usage Guide 1.0*
280 DMTF [DSP1012](#), *Boot Control Profile 1.0*
281 DMTF [DSP1022](#), *CPU Profile 1.0*
282 DMTF [DSP1027](#), *Power State Management Profile 1.0*
283 DMTF [DSP1033](#), *Profile Registration Profile 1.0*
284 DMTF [DSP1041](#), *Resource Allocation Profile 1.0*
285 DMTF [DSP1043](#), *Allocation Capabilities Profile 1.0*
286 DMTF [DSP1052](#), *Computer System Profile 1.0*
287 DMTF [DSP1057](#), *Virtual System Profile 1.0*

288 2.2 References under Development

- 289 DMTF [DSP1044](#), *Processor Device Resource Virtualization Profile 0.7*
290 DMTF [DSP1045](#), *Memory Resource Virtualization Profile 0.7*
291 DMTF [DSP1047](#), *Block Based Storage Resource Virtualization Profile 0.2*
292 DMTF [DSP1048](#), *Memory Resource Virtualization Profile 0.7*
293 DMTF [DSP1049](#), *Storage Adapter Resource Virtualization Profile 0.7*
294 DMTF [DSP1059](#), *Generic Device Resource Virtualization Profile 0.9*

295 **2.3 Other References**

296 ISO/IEC [Directives, Part 2](#), *Rules for the structure and drafting of International Standards*

297 OMG [UMLINF2.0](#), *Unified Modeling Language: Infrastructure*

298 OMG [UMLSUP2.0](#), *Unified Modeling Language: Superstructure*

299 OPENSLLP [RFC2608](#), *RFC Service Location Protocol Version 2*

300 **3 Terms and Definitions**

301 For the purposes of this document, the following terms and definitions and the terms defined in DSP1001
302 apply.

303 **3.1**

304 **can**

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

306 **3.2**

307 **cannot**

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

309 **3.3**

310 **conditional**

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

313 **3.4**

314 **mandatory**

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

317 **3.5**

318 **may**

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

320 **3.6**

321 **need not**

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

323 **3.7**

324 **optional**

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

326 **3.8**

327 **referencing profile**

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

- 330 **3.9**
331 **shall**
332 indicates requirements to be followed strictly in order to conform to the document and from which no
333 deviation is permitted
- 334 **3.10**
335 **shall not**
336 indicates requirements to be followed strictly in order to conform to the document and from which no
337 deviation is permitted
- 338 **3.11**
339 **should**
340 indicates that among several possibilities, one is recommended as particularly suitable, without mention-
341 ing or excluding others, or that a certain course of action is preferred but not necessarily required
- 342 **3.12**
343 **should not**
344 indicates that a certain possibility or course of action is deprecated but not prohibited
- 345 **3.13**
346 **unspecified**
347 indicates that this profile does not define any constraints for the referenced CIM element
- 348 **3.14**
349 **implementation**
350 a set of software components that realize the classes that are specified or specialized by this profile
- 351 **3.15**
352 **client**
353 application that exploits facilities specified by this profile
- 354 **3.16**
355 **virtualization platform**
356 virtualizing infrastructure provided by a host system that enables the deployment of virtual systems

357 **4 Symbols and Abbreviated Terms**

358 The following symbols and abbreviations are used in this document.

- 359 **4.1**
360 **CIMOM**
361 CIM object manager
- 362 **4.2**
363 **RASD**
364 resource allocation setting data
- 365 **4.3**
366 **SLP**
367 service location protocol

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

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

374 **5 Synopsis**

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

376 **Version:** 1.0.0a

377 **Organization:** DMTF

378 **CIM Schema Version:** 2.16

379 **Central Class:** CIM_System

380 **Scoping Class:** CIM_System

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

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

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

389

Table 1 – Related Profiles

Profile Name	Organization	Version	Relationship	Description
<i>Profile Registration</i>	DMTF	1.0	Mandatory	The DMTF management profile that describes the registration of DMTF management profiles; see 7.2.
<i>Virtual System</i>	DMTF	1.0	Mandatory	The autonomous DMTF management profile that specifies the minimum object model needed for the inspection and basic manipulation of a virtual system; see 7.3.
<i>Processor Device Resource Virtualization</i>	DMTF	1.0	Conditional	The component DMTF management profile that specifies the allocation of processor resources; see 7.2.2.
<i>Memory Resource Virtualization</i>	DMTF	1.0	Conditional	The component DMTF management profile that specifies the allocation of memory resources; see 7.2.2.
<i>Storage Adapter Resource Virtualization</i>	DMTF	1.0	Conditional	The component DMTF management profile that specifies the allocation of storage adapter resources; see 7.2.2.

Profile Name	Organization	Version	Relationship	Description
<i>Generic Device Resource Virtualization</i>	DMTF	1.0	Conditional	The component DMTF management profile that specifies the allocation of generic resources; see 7.2.2.

390 6 Description (Informative)

391 This profile defines a top-level object model for the inspection and control of system virtualization facilities
392 provided by host systems. It supports the following range of functions:

- 393 • the detection of host systems that provide system virtualization facilities
- 394 • the discovery of scoped host resources
- 395 • the discovery of scoped resource pools
- 396 • the inspection of host system capabilities for
 - 397 – the creation and manipulation of virtual systems
 - 398 – the allocation of resources of various types
- 399 • the inspection of resource pool capabilities
- 400 • the discovery of hosted virtual systems
- 401 • the inspection of relationships between host entities (host systems, host resources, and re-
402 source pools) and virtual entities (virtual systems and virtual resources)
- 403 • the creation and manipulation of virtual systems using input configurations, predefined
404 configurations available at the host system, or both
- 405 • the creation and manipulation of snapshots that capture the configuration and state of a virtual
406 system at a particular point in time

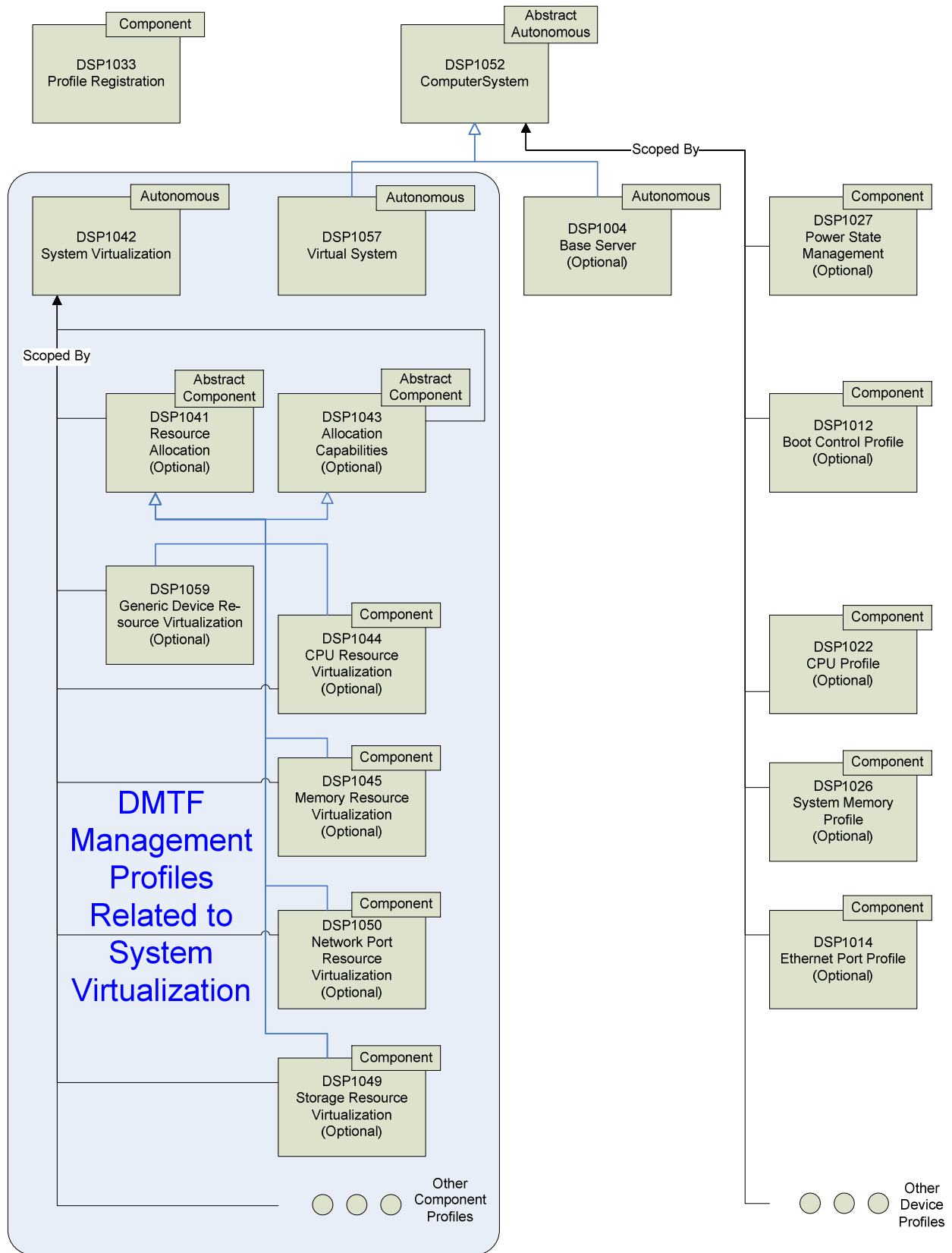
407 6.1 DMTF Management Profile Relationships

408 A client that is exploiting system virtualization facilities specified by the *System Virtualization Profile* needs
409 to be virtualization aware. The specified model keeps that knowledge at an abstract level that is
410 independent of a particular system virtualization platform implementation or technology.

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

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

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



424

425

Figure 1 – DMTF Management Profiles Related to System Virtualization

426 For example, an implementation that instruments a virtualization platform may implement some of the fol-
427 lowing DMTF management profiles:

- 428 • *System Virtualization Profile*

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

- 431 • *Virtual System Profile*

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

- 433 • Resource-type-specific profiles

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

- 438 • Resource allocation profiles

439 Resource allocation profiles enable the inspection and management of resource allocation re-
440 quests, allocated resources, and resources available for allocation. Resource allocation profiles
441 are based on the abstract *Resource Allocation Profile* and on the abstract *Allocation*
442 *Capabilities Profile*. Resource allocation profiles are scoped by the *System Virtualization Profile*.
443 A client may exploit resource allocation profiles for the inspection of

- 444 – allocated resources

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

- 446 – capabilities that describe possible values for allocation requests

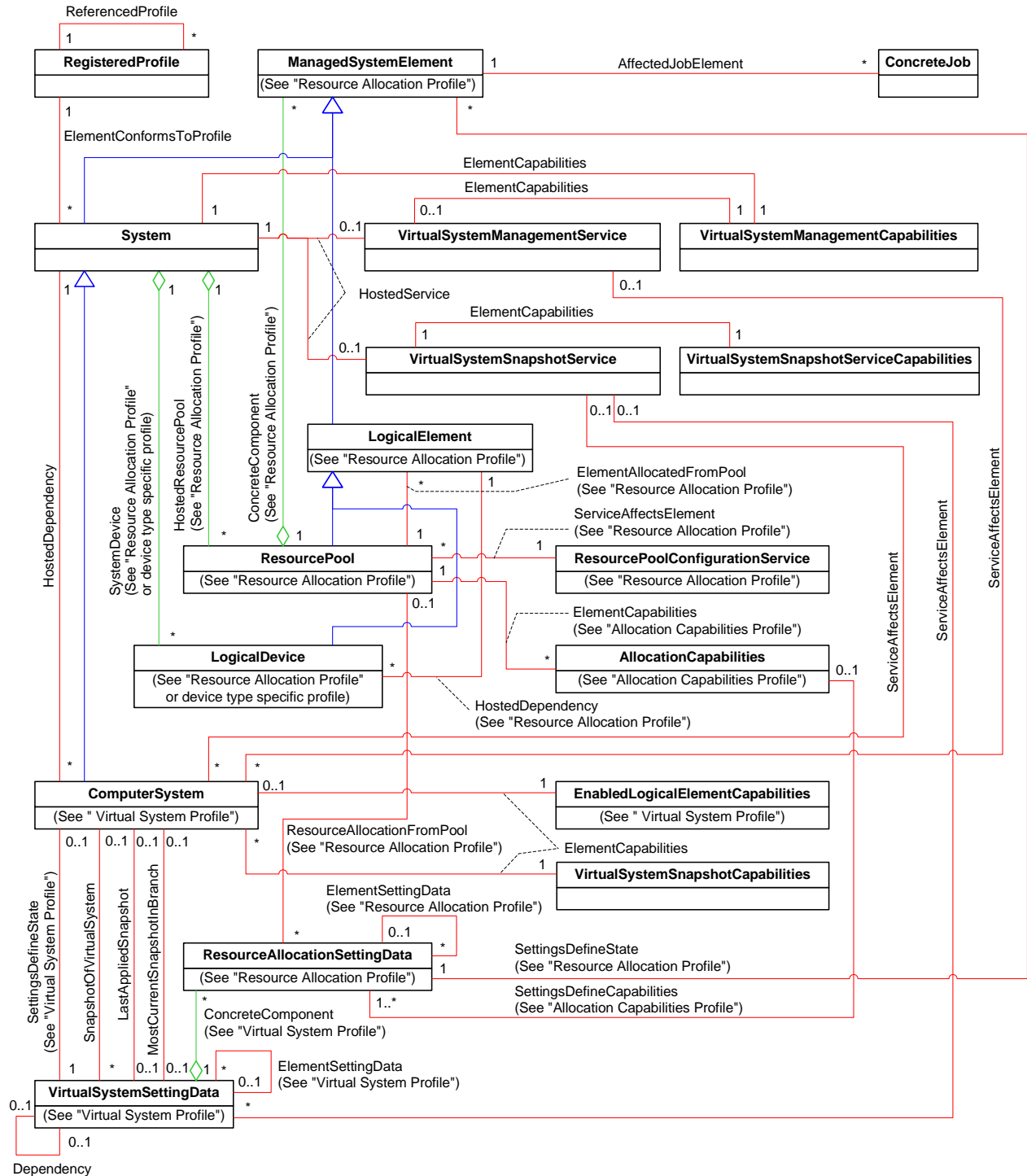
- 447 – capabilities that describe the mutability of resource allocations

448 For some resource types, specific resource allocation profiles are specified that address re-
449 source-type-specific resource allocation aspects and capabilities. Examples are the *Processor*
450 *Device Resource Virtualization Profile* and the *Storage Adapter Resource Virtualization Profile*.

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

454 6.2 System Virtualization Class Schema

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



459
460

461 **Figure 2 – System Virtualization Profile: Class Diagram**

462 The *System Virtualization Profile* specifies the use of the following classes and associations:

- 463 • the *CIM_RegisteredProfile* class and the *CIM_ElementConformsToProfile* association for the
- 464 advertisement of conformance to this profile

- 465 • the CIM_ReferencedProfile association for the representation of a scoping relationship between
466 this profile and scoped DMTF management profiles
- 467 • the CIM_System class for the representation of host systems
- 468 • the CIM_HostedDependency association for the representation of the hosting relationship be-
469 tween a host system and hosted virtual systems
- 470 • the CIM_VirtualSystemManagementService class for the representation of virtual system
471 management services available at a host system, providing operations like the creation and
472 modification of virtual systems and their components
- 473 • the CIM_HostedService association for the representation of the relationship between a host
474 system and services that it provides
- 475 • the CIM_VirtualSystemManagementCapabilities class for the representation of optional fea-
476 tures, properties, and methods available for the management of virtual systems hosted by a
477 host system
- 478 • the CIM_ElementCapabilities association for the representation of the relationship between a
479 host system, a virtual system or a service, and their respective capabilities
- 480 • the CIM_ServiceAffectsElement association for the representation of the relationship between
481 defined services and affected elements like virtual systems or virtual system snapshots
- 482 • the CIM_VirtualSystemSettingData class for the representation of snapshots (in addition to the
483 use of that class for the representation of virtual aspects of a virtual system as specified by the
484 *Virtual System Profile*)
- 485 • the CIM_VirtualSystemSnapshotService class for the representation of snapshot-related ser-
486 vices available at a host system
- 487 • the CIM_VirtualSystemSnapshotServiceCapabilities class for the representation of optional fea-
488 tures, properties, and methods available for the management of snapshots of virtual systems
- 489 • the CIM_VirtualSystemSnapshotCapabilities class for the representation of optional features,
490 properties, and methods available for the management of snapshots relating to one particular
491 virtual system
- 492 • the CIM_SnapshotOfVirtualSystem association for the representation of the relationship be-
493 tween a snapshot of a virtual system and the virtual system itself
- 494 • the CIM_Dependency association for dependencies among virtual system snapshots
- 495 • the CIM_LastAppliedSnapshot association for the representation of the relationship between a
496 virtual system and the snapshot that was most recently applied to it
- 497 • the CIM_MostCurrentSnapshotInBranch association for the representation of the relationship
498 between a virtual system and the snapshot that is the most current snapshot in a sequence of
499 snapshots captured from the virtual system
- 500 • the CIM_ConcreteJob class and the CIM_AffectedJobElement association to model a mecha-
501 nism that allows tracking of asynchronous tasks resulting from operations such as the optional
502 CreateSystem() method of the CIM_VirtualSystemManagementService class

503 In general, any mention of a class in this document means the class itself or its subclasses. For example,
504 a statement such as “an instance of the CIM_LogicalDevice class” implies an instance of the CIM_Logi-
505 calDevice class or a subclass of the CIM_LogicalDevice class.

506 **6.3 Virtual System Configurations**

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

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

511 The *Virtual System Profile* defines the concept of virtual system configurations and applies it to the follow-
512 ing types of virtual system configurations:

- 513 • the “State” virtual system configuration, which represents a virtualization-specific state that ex-
514 tends a virtual system representation
- 515 • the “Defined” virtual system configuration, which represents virtual system definitions
- 516 • the “Next” virtual system configuration, which represents the virtual system configuration that
517 will be used for the next activation of a virtual system

518 The *System Virtualization Profile* applies the concept of virtual system configurations and defines the
519 following additional types of virtual system configurations:

- 520 • the “Input” virtual system configuration, which represents configuration information for new vir-
521 tual systems
- 522 • the “Reference” virtual system configuration, which represents configuration information that
523 complements an “Input” virtual system configuration for a new virtual system
- 524 • the “Snapshot” virtual system configuration, which represents snapshots of virtual systems

525 6.4 Resource Allocation

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

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

532 Resource allocation is the process of obtaining an allocated resource based on a resource allocation re-
533 quest. This profile distinguishes two types of resource allocation:

- 534 • Persistent Resource Allocation

535 Persistent resource allocation occurs while virtual resources are defined and supporting re-
536 sources are persistently allocated from a resource pool.

- 537 • Transient Resource Allocation

538 Transient resource allocation occurs as virtual resources are instantiated and supporting re-
539 sources are temporarily allocated from a resource pool for the lifetime of the virtual resource in-
540 stance.

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

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

544 EXAMPLE 2: Transient Resource Allocation: Host memory

545 A contiguous chunk of host memory is temporarily allocated to support virtual memory as the scoping vir-
546 tual system is instantiated. The memory chunk remains allocated for the time that the virtual system
547 remains instantiated.

548 EXAMPLE 3: Transient Resource Allocation: I/O bandwidth

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

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

556 **6.5 Snapshots**

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

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

567 Implementations may establish relationships between snapshots. For example, snapshots may be or-
568 dered by their creation time.

569 The *System Virtualization Profile* specifies mechanisms for the creation, application, and destruction of
570 snapshots. It specifies a snapshot model that enables the inspection of snapshot-related configuration
571 information such as the virtual system configurations that were effective when the snapshot was captured.
572 Relationships between snapshots are also modeled.

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

576 **7 Implementation**

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

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

581 Where reference is made to CIM Schema properties that enumerate values, the numeric value is norma-
582 tive and the descriptive text following it in parentheses is informational. For example, in the statement "If
583 an instance of the CIM_VirtualSystemManagementCapabilities class contains the value
584 3 (DestroySystemSupported) in an element of the SynchronousMethodsSupported[] array property," the
585 value "3" is normative text and "(DestroySystemSupported)" is informational text.

586 **7.1 Host System**

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

589 **7.2 Profile Registration**

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

592 7.2.1 System Virtualization Profile

593 The implementation of the *System Virtualization Profile* shall be indicated by an instance of the
 594 CIM_RegisteredProfile class in the CIM Interop namespace. Each instance of the CIM_System class that
 595 represents a host system that is manageable through this profile shall be a central instance of this profile
 596 by associating it with the instance of the CIM_RegisteredProfile class through an instance of the
 597 CIM_ElementConformsToProfile association.

598 7.2.2 Scoped Resource Allocation Profiles (Conditional)

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

601 The support of scoped resource-allocation profiles is conditional with respect to the presence of an in-
 602 stance of the CIM_RegisteredProfile class in the Interop namespace that represents the scoped resour-
 603 ce-allocation profile implementation and is associated with the instance of the CIM_RegisteredProfile
 604 class that represents an implementation of the *System Virtualization Profile* through an instance of the
 605 CIM_ReferencedProfile association.

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

609 An implementation that provides conditional support for inspecting and managing the allocation of re-
 610 sources of one particular resource type shall apply one of the following implementation approaches:

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

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

- 617 • The instance of the CIM_RegisteredProfile class that represents the implementation of this pro-
 618 file and the instance of the CIM_RegisteredProfile class that represents the implementation of
 619 the scoped resource-allocation DMTF management profile shall be associated through an in-
 620 stance of the CIM_ReferencedProfile association.
- 621 • One of the following conditions regarding profile implementation advertisement shall be met:
 - 622 – Central Class Profile Implementation Advertisement:
 623 Instances of the CIM_ElementConformsToProfile association shall associate each instance
 624 of the CIM_ResourcePool class that is a central instance of the scoped-resource-allocation
 625 DMTF management profile with the instance of the CIM_RegisteredProfile class that repre-
 626 sents an implementation of the scoped-resource-allocation DMTF management profile.
 - 627 – Scoping Class Profile Implementation Advertisement:
 628 No instances of the CIM_ElementConformsToProfile association shall associate any in-
 629 stance of the CIM_ResourcePool class that is a central instance of the scoped-resource-
 630 allocation DMTF management profile with the instance of the CIM_RegisteredProfile class
 631 that represents an implementation of the scoped-resource-allocation DMTF management
 632 profile.

633 7.3 Representation of Hosted Virtual Systems

634 The *System Virtualization Profile* strengthens the requirements for the representation of virtual system
 635 configurations specified by the *Virtual System Profile* for hosted virtual systems.

636 **7.3.1 Profile Conformance for Hosted Virtual Systems**

637 Any virtual system that is hosted by a conformant host system shall be represented by an instance of the
638 CIM_ComputerSystem class that is a central instance of the *Virtual System Profile*. That instance shall be
639 associated with the instance of the CIM_System class that represents the conformant host system
640 through an instance of the CIM_HostedDependency association.

641 **7.3.2 CIM_VirtualSystemSettingData.VirtualSystemType Property**

642 The value of the VirtualSystemType property shall be equal to an element of the
643 VirtualSystemTypesSupported[] array property in the instance of the
644 CIM_VirtualSystemManagementCapabilities class that is associated with the instance of the
645 CIM_VirtualSystemManagementService class that represents the host system, or shall be NULL if the
646 value of the VirtualSystemTypesSupported[] array property is NULL (see 7.4.2).

647 **7.4 Virtual System Management Capabilities**

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

650 **7.4.1 CIM_VirtualSystemManagementCapabilities Class**

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

654 **7.4.2 CIM_VirtualSystemManagementCapabilities.VirtualSystemTypesSupported[] 655 Array Property (Optional)**

656 The VirtualSystemTypesSupported[] array property should be supported. Array values shall designate the
657 set of supported virtual system types. If the VirtualSystemTypesSupported[] array property is not imple-
658 mented (has a value of NULL), the implementation does not externalize the set of supported virtual
659 system types, but internally still may exhibit different types of virtual systems.

660 **7.4.3 CIM_VirtualSystemManagementCapabilities.SynchronousMethodsSupported[] 661 Array Property (Optional)**

662 The SynchronousMethodsSupported[] array property should be supported. Array values shall designate
663 the set of methods of the CIM_VirtualSystemManagementService class that are supported with synchro-
664 nous behavior only. A NULL value or an empty value set shall be used to indicate that no methods are
665 supported with synchronous behavior. If a method is designated within the value set of the
666 SynchronousMethodsSupported[] property, that method shall always exhibit synchronous behavior and
667 shall not be designated within the value set of the AsynchronousMethodsSupported[] property.

668 **7.4.4 CIM_VirtualSystemManagementCapabilities.AsynchronousMethodsSupported[] 669 Array Property (Optional)**

670 The AsynchronousMethodsSupported[] array property should be supported. Array values shall designate
671 the set of methods of the CIM_VirtualSystemManagementService class that are supported with synchro-
672 nous and potentially with asynchronous behavior. A NULL value or an empty value set shall be used to
673 indicate that no methods are supported with asynchronous behavior. If a method is designated with a
674 value in the AsynchronousMethodsSupported[] array property, it may show either synchronous or
675 asynchronous behavior.

676 **7.4.5 CIM_VirtualSystemManagementCapabilities.IndicationsSupported[] Array**
677 **Property (Optional)**

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

681 **7.4.6 Grouping Rules for Implementations of Methods of the**
682 **CIM_VirtualSystemManagementService Class**

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

686 **7.4.6.1 Support of Virtual System Definition and Destruction**

687 If virtual system definition and destruction are supported, the DefineSystem() and DestroySystem()
688 methods of the CIM_VirtualSystemManagementService class shall be implemented, and the values
689 2 (DefineSystemSupported) and 3 (DestroySystemSupported) shall be set in the
690 SynchronousMethodsSupported[] or AsynchronousMethodsSupported[] array properties within the
691 instance of the CIM_VirtualSystemManagementCapabilities class that describes capabilities of the imple-
692 mentation.

693 If virtual system definition and destruction are not supported, the values 2 (DefineSystemSupported) and
694 3 (DestroySystemSupported) shall not be set in the SynchronousMethodsSupported[] or
695 AsynchronousMethodsSupported[] array properties of the instance of the
696 CIM_VirtualSystemManagementCapabilities class that describes the virtual system management capabili-
697 ties of the host system.

698 **7.4.6.2 Support of Virtual Resource Addition and Removal**

699 If the addition and removal of virtual resources to or from virtual systems is supported, the
700 AddResourceSettings() and RemoveResourceSettings() methods of the
701 CIM_VirtualSystemManagementService class shall be implemented, and the values
702 1 (AddResourceSettingsSupported) and 7 (RemoveResourceSettingsSupported) shall be set in the
703 SynchronousMethodsSupported[] or AsynchronousMethodsSupported[] array properties of the instance
704 of the CIM_VirtualSystemManagementCapabilities class that describes the virtual system management
705 capabilities of the host system.

706 If the addition and removal of virtual resources to virtual systems is not supported, the values
707 1 (AddResourceSettingsSupported) and 7 (RemoveResourceSettingsSupported) shall not be set in the
708 SynchronousMethodsSupported[] or AsynchronousMethodsSupported[] array properties of the instance
709 of the CIM_VirtualSystemManagementCapabilities class that describes the virtual system management
710 capabilities of the host system.

711 **7.4.6.3 Support of Virtual System and Resource Modification**

712 If the modification of virtual systems and virtual resources is supported, the ModifyResourceSettings()
713 and ModifySystemSettings() methods of the CIM_VirtualSystemManagementService class shall be
714 implemented, and the values 5 (ModifyResourceSettingsSupported) and
715 6 (ModifySystemSettingsSupported) shall be set in the SynchronousMethodsSupported[] or
716 AsynchronousMethodsSupported[] array properties of the instance of the
717 CIM_VirtualSystemManagementCapabilities class that describes the virtual system management capabili-
718 ties of the host system.

719 If the modification of virtual systems and virtual resources is not supported, the values
720 5 (ModifyResourceSettingsSupported) and 6 (ModifySystemSettingsSupported) shall not be set in the
721 SynchronousMethodsSupported[] or AsynchronousMethodsSupported[] array properties of the instance

722 of the `CIM_VirtualSystemManagementCapabilities` class that describes the virtual system management
723 capabilities of the host system.

724 **7.5 Virtual System Definition and Modification**

725 The *System Virtualization Profile* specifies methods for the definition and modification of virtual systems.
726 These method specifications use the `CIM_VirtualSystemSettingData` class for the parameterization of
727 system-specific properties. Subsequent subclauses specify:

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

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

736 **7.5.1 CIM_VirtualSystemSettingData.InstanceID Property**

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

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

743 An implementation shall use a non-`NULL` value to identify an existing instance of the
744 `CIM_VirtualSystemSettingData` class. If the value does not identify an instance of the
745 `CIM_VirtualSystemSettingData` class, an implementation shall return a return code that indicates an inva-
746 lid parameter (see 8.2.4.3).

747 **7.5.2 CIM_VirtualSystemSettingData.ElementName Property (Optional)**

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

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

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

756 **7.5.3 CIM_VirtualSystemSettingData.VirtualSystemIdentifier Property (Optional)**

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

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

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

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

766 **7.5.4 CIM_VirtualSystemSettingData.VirtualSystemType Property (Optional)**

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

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

777 **7.6 Virtual Resource Definition and Modification**

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

787 **7.7 Virtual System Snapshots (Optional)**

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

789 **7.7.1 Virtual System Snapshot Service and Capabilities**

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

792 **7.7.1.1 Support of Virtual System Snapshots (Optional)**

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

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

- 796 • the CIM_VirtualSystemSnapshotService class shall be implemented and the following methods
 797 shall be supported:
 - 798 – CreateSnapshot(), for at least one type of snapshot
 - 799 – DestroySnapshot()
 - 800 – ApplySnapshot()
- 801 • There shall be exactly one instance of the CIM_VirtualSystemSnapshotService class associated
 802 to the central instance of this profile through an instance of the CIM_HostedService association.

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

805 **7.7.1.2 CIM_VirtualSystemSnapshotServiceCapabilities Class (Conditional)**

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

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

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

- 815 • 2 (CreateSnapshotSupported)
- 816 • 3 (DestroySnapshotSupported)
- 817 • 4 (ApplySnapshotSupported)

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

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

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

830 **7.7.2 Virtual System Snapshot Representation (Conditional)**

831 Virtual snapshot representation is conditional with respect to the support of virtual system snapshots (see
832 7.7.1.1).

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

837 **7.7.3 Designation of the Last Applied Snapshot (Conditional)**

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

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

844 **7.7.4 Designation of the Most Current Snapshot in Branch (Conditional)**

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

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

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

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

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

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

854 7.7.5 Virtual System Snapshot Capabilities (Optional)

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

857 7.7.5.1 CIM_VirtualSystemSnapshotCapabilities.SnapshotTypesEnabled[] Array Property

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

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

863 7.7.5.2 CIM_VirtualSystemSnapshotCapabilities.GuestOSNotificationEnabled Property 864 (Optional)

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

870 8 Methods

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

873 8.1 General Behavior of Extrinsic Methods

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

875 8.1.1 Resource Allocation Requests

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

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

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

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

891 **8.1.2 Method Results**

892 If the implementation does not support a method, it shall set a return value of 1 (Not Supported).

893 If synchronous execution of a method succeeds, the implementation shall set a return value of
894 0 (Completed with No Error).

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

897 If a method is executed as an asynchronous task, the implementation shall perform all of the following ac-
898 tions:

- 899 • Set a return value of 4096 (Job Started).
- 900 • Set the value of the Job output parameter to refer to an instance of the CIM_ConcreteJob class
901 that represents the asynchronous task.
- 902 • Set the values of the JobState and TimeOfLastStateChange properties in that instance to repre-
903 sent the state and last state change time of the asynchronous task.

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

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

- 907 • Set the value of the JobState property to 7 (Completed).
- 908 • Provide an instance of the CIM_AffectedJobEntity association with property values set as fol-
909 lows:
 - 910 – The value of the AffectedElement property shall refer to the object that represents the top-
911 level entity that was created or modified by the asynchronous task. For example, for the
912 DefineSystem() method, this is an instance of the CIM_ComputerSystem class, and for
913 the CreateSnapshot() method, this is an instance of the CIM_VirtualSystemSettingData
914 class that represents a snapshot of a virtual system.
 - 915 – The value of the AffectingElement property shall refer to the instance of the
916 CIM_ConcreteJob class that represents the completed asynchronous task.
 - 917 – The value of the first element in the ElementEffects[] array property (ElementEffects[0])
918 shall be set to 5 (Create) for the DefineSystem() or CreateSnapshot() methods. Other-
919 wise, this value shall be 0 (Unknown).

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

922 **8.1.3 Asynchronous Processing**

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

925 **8.1.3.1 General Requirements**

926 All of the following conditions shall be met:

- 927 • Elements that convey information about which methods of the
928 CIM_VirtualSystemManagementService class are supported for asynchronous execution within
929 an implementation are modeled in 7.4.4.
- 930 • Elements that convey information about which methods of the
931 CIM_VirtualSystemSnapshotService class are supported for asynchronous execution within an
932 implementation are modeled in 7.7.1.1.
- 933 • Elements that convey information about whether a method is executed asynchronously are
934 modeled in 8.1.2.

935 **8.1.3.2 Job Parameter**

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

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

942 **8.2 Methods of the VirtualSystemManagementService Class**

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

945 **8.2.1 CIM_VirtualSystemManagementService.DefineSystem() Method (Conditional)**

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

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

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

954 Table 2 contains requirements for parameters of this method.

955 **Table 2 – DefineSystem() Method: Parameters**

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

956 8.2.1.1 Value Preference Rules

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

- 959 • “Input” virtual system configuration

960 The “Input” virtual system configuration is prepared locally by the client and provided in the form
961 of embedded instances of the CIM_VirtualSystemSettingData class in the SystemSettings pa-
962 rameter and embedded instances of the CIM_ResourceAllocationSettingData class as values
963 for elements of the ResourceSettings[] array parameter.

- 964 • “Reference” virtual system configuration

965 The “Reference” virtual system configuration is a “Defined” virtual system configuration that al-
966 ready exists within the implementation; it is referenced by the ReferencedConfiguration
967 parameter.

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

971 If only the “Reference” virtual system configuration is provided by the client, the implementation shall cre-
972 ate a copy or cloned configuration of the “Reference” virtual system configuration.

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

- 976 • The basic level supports the addition of resource allocations that were not requested by ele-
977 ments of the ResourceSettings[] array parameter, but that are defined in the “Reference” virtual
978 system configuration.
- 979 • The advanced level, in addition, supports amending incomplete resource requests.

980 In this case the correlation of instances of the CIM_ResourceAllocationSettingData class in the
981 “Input” configuration and in the “Reference” configuration shall be established through the value
982 of the InstanceID parameter. If the value of the InstanceID parameter is identical for an instance
983 in the “Input” configuration and an instance in the “Reference” configuration, these instances to-
984 gether describe one virtual resource allocation request, such that non-NULL property values
985 specified in the “Input” configuration override those specified in the “Reference” configuration.

986 If no value is specified for a property in the “Input” configuration or in the “Reference” configuration, the
987 implementation may exhibit an implementation-dependent default behavior. The *Resource Allocation*
988 *Profile*, the *Generic Device Resource Virtualization Profile*, and resource-type-specific resource allocation
989 DMTF management profiles may specify resource-type-specific behavior.

990 If the DefineSystem() method is called without input parameters, the implementation may exploit a de-
991 fault behavior or may fail the method execution.

992 NOTE: A client may inspect the “Reference” virtual system configuration before invoking the DefineSystem()
993 method (see respective use cases in the *Virtual System Profile*).

994 8.2.1.2 SystemSettings Parameter

995 A client should set the value of the SystemSettings parameter with an embedded instance of the
996 CIM_VirtualSystemSettingData class that describes requested virtual system settings. The client may set
997 the value of the SystemSettings parameter to NULL, requesting the implementation to select input values
998 based on the rules specified in 8.2.1.1.

999 An implementation shall interpret the value of the SystemSettings parameter as the system part of an
1000 “Input” virtual system configuration, and apply the rules specified in 8.2.1.1.

1001 The use of the CIM_VirtualSystemSettingData class as input for operations specified by this profile is
 1002 specified in 10.22.

1003 **8.2.1.3 ResourceSettings[] Array Parameter**

1004 A client should set the ResourceSettings[] array parameter and apply the specifications given in 8.1.1.
 1005 The client may set the value of the ResourceSettings[] array parameter to NULL or provide an empty ar-
 1006 ray, requesting the implementation to define a default set of virtual resources (see 8.2.1.1).

1007 An implementation shall interpret the value of the ResourceSettings[] array parameter as the resource
 1008 part of an “Input” virtual system configuration, and apply the value preference rules specified in 8.2.1.1.

1009 **8.2.1.4 ReferencedConfiguration Parameter**

1010 A client may set a value of the ReferencedConfiguration parameter to refer to an existing “Defined” virtual
 1011 system configuration. A client may set the value of the ReferencedConfiguration parameter to NULL, indi-
 1012 cating that a “Reference” configuration shall not be used.

1013 An implementation shall use the “Reference” virtual system configuration according to the rules specified
 1014 in 8.2.1.1.

1015 **8.2.1.5 ResultingSystem Parameter**

1016 The implementation shall set the value of the ResultingSystem parameter as follows:

- 1017 • If the method execution is performed synchronously and is successful, the value is set to refer-
 1018 ence the instance of the CIM_ComputerSystem class that represents the newly defined virtual
 1019 system.
- 1020 • If the method execution is performed synchronously and fails, or if the method execution is per-
 1021 formed asynchronously, the value is set to NULL.

1022 **8.2.1.6 Return Codes**

1023 An implementation shall indicate the result of the method execution by using the return code values speci-
 1024 fied in Table 3.

1025 **Table 3 – DefineSystem() Method: Return Code Values**

Value	Description
0	Method execution was successful.
1	Method is not supported.
2	Method execution failed.
3	Method execution failed because a timeout condition occurred.
4	Method execution failed because invalid parameters were specified by the client.
4096	Method execution is performed asynchronously. The specifications given in 8.1.3 apply.

1026 **8.2.2 CIM_VirtualSystemManagementService.DestroySystem() Method (Conditional)**

1027 Behavior applicable to all extrinsic methods is specified in 8.1.2.

1028 The implementation of the DestroySystem() method is conditional with respect to the support of the
 1029 definition and destruction of virtual systems (see 7.4.6.1).

1030 The execution of the DestroySystem() method shall effect the destruction of the referenced virtual system
 1031 and all related virtual system configurations, including snapshots.

1032 Table 4 contains requirements for parameters of this method.

1033 **Table 4 – DestroySystem() Method: Parameters**

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

1034 **8.2.2.1 AffectedSystem Parameter**

1035 A client shall set a value of the AffectedSystem parameter to refer to the instance of the
1036 CIM_ComputerSystem class that represents the virtual system to be destroyed.

1037 An implementation shall interpret the value of the AffectedSystem parameter to identify the virtual system
1038 that is to be destroyed.

1039 **8.2.2.2 Return Codes**

1040 An implementation shall indicate the result of the method execution by using the return code values speci-
1041 fied in Table 5.

1042 **Table 5 – DestroySystem() Method: Return Code Values**

Value	Description
0	Method execution was successful.
1	Method is not supported.
2	Method execution failed.
3	Method execution failed because a timeout condition occurred.
4	Method execution failed because the system could not be found.
5	Method execution failed because the affected system is in a state in which the implementation rejects destruction.
4096	Method execution is performed asynchronously. The specifications given in 8.1.3 apply.

1043 **8.2.3 CIM_VirtualSystemManagementService.AddResourceSettings() Method**
1044 **(Conditional)**

1045 Behavior applicable to all extrinsic methods is specified in 8.1.2.

1046 The implementation of the AddResourceSettings() method is conditional with respect to the support of
1047 the addition and the removal of virtual resources to virtual systems (see 7.4.6.2).

1048 The execution of the AddResourceSettings() method shall effect the entry of resource allocation requests
1049 or resource allocations provided through the ResourceSettings[] array parameter in the affected virtual
1050 system configuration.

1051 Table 6 contains requirements for parameters of this method.

1052 **Table 6 – AddResourceSettings() Method: Parameters**

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

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

1053 **8.2.3.1 AffectedConfiguration Parameter**

1054 A client shall set a value of AffectedConfiguration parameter to refer to the instance of the
 1055 CIM_VirtualSystemSettingData class that represents the virtual system configuration that receives new
 1056 resource allocations.

1057 An implementation shall interpret the value of the AffectedConfiguration parameter to identify the virtual
 1058 system configuration that receives new resource allocations.

1059 **8.2.3.2 ResourceSettings[] Array Parameter**

1060 A client shall set the ResourceSettings[] parameter.

1061 An implementation shall apply the specifications given in 8.1.1.

1062 **8.2.3.3 ResultingResourceSettings[] Array Parameter**

1063 The implementation shall set the value of the ResultingResourceSettings[] array parameter as follows:

- 1064 • to an array of references to instances of the CIM_ResourceAllocationSettingData class that
 1065 represent resource allocations that were obtained during the execution of the method
- 1066 • to NULL, if the method is executed synchronously and fails, or if the method is executed
 1067 asynchronously

1068 **8.2.3.4 Return Codes**

1069 An implementation shall indicate the result of the method execution by using the return code values speci-
 1070 fied in Table 7.

1071 **Table 7 – AddResourceSettings() Method: Return Code Values**

Value	Description
0	Method execution was successful.
1	Method is not supported.
2	Method execution failed.
3	Method execution failed because a timeout condition occurred.
4	Method execution failed because invalid parameters were specified by the client.
4096	Method execution is performed asynchronously. The specifications given in 8.1.3 apply.

1072 **8.2.4 CIM_VirtualSystemManagementService.ModifyResourceSettings() Method**
 1073 **(Conditional)**

1074 Behavior applicable to all extrinsic methods is specified in 8.1.2.

1075 The implementation of the `ModifyResourceSettings()` method is conditional with respect to support of the
1076 modification of virtual systems and resources (see 7.4.6.3).

1077 If implemented, the execution of the `ModifyResourceSettings()` method shall effect the modification of re-
1078 source allocation requests that exist, with the implementation using instances of the
1079 `CIM_ResourceAllocationSettingData` class that are passed in through values of elements of the
1080 `ResourceSettings[]` array parameter.

1081 The execution of the `ModifyResourceSettings()` method shall effect the modification of resource alloca-
1082 tions or resource allocation requests, such that non-key and non-NULL values of instances of the
1083 `CIM_ResourceAllocationSettingData` class provided as values for elements of the `ResourceSettings[]` ar-
1084 ray parameter override respective values in instances identified through the `InstanceID` property.

1085 Table 8 contains requirements for parameters of this method.

1086 **Table 8 – `ModifyResourceSettings()` Method: Parameters**

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

1087 **8.2.4.1 `ResourceSettings[]` Parameter**

1088 The specifications in 8.1.1 apply.

1089 A client shall set the `ResourceSettings[]` parameter. Any instance of the
1090 `CIM_ResourceAllocationSettingData` class that is passed in as a value for elements of the
1091 `ResourceSettings[]` array parameter shall conform to all of the following conditions:

- 1092 • It shall represent requests for the modification of virtual resource state extensions, virtual re-
1093 source definitions scoped by one particular virtual system, or both.
- 1094 • It shall have a valid non-NULL value in the `InstanceID` property that identifies a respective in-
1095 stance of the `CIM_ResourceAllocationSettingData` class that represents an existing resource
1096 allocation or resource allocation request within the implementation. This should be assured
1097 through the execution of previously executed retrieve operations, such as the execution of
1098 extrinsic methods or intrinsic CIM operations that yield respective instances of the
1099 `CIM_ResourceAllocationSettingData` class. For example, the client may use the intrinsic
1100 `GetInstance()` CIM operation.

1101 The client shall modify such instances locally to reflect the desired modifications and finally pass
1102 them back in as elements of the `ResourceSettings[]` array parameter. Modifications shall not be
1103 applied to the `InstanceID` property that is the key property of the
1104 `CIM_ResourceAllocationSettingData` class. Further restriction may apply, such as from re-
1105 source-type-specific resource allocation DMTF management profiles.

1106 An implementation shall apply the specifications given in 8.1.1. The implementation shall ignore any ele-
1107 ment of the `ResourceSettings[]` array property that does not identify, through the value of the `InstanceID`
1108 key property, an existing instance of the `CIM_ResourceAllocationSettingData` class within the
1109 implementation.

1110 **8.2.4.2 `ResultingResourceSettings[]` Parameter**

1111 The implementation shall set the value of the `ResultingResourceSettings[]` array parameter as follows:

- 1112 • If the method was executed asynchronously, the value shall be set to NULL.

- 1113 • If the method was executed synchronously and one or more resources were successfully modified, for each successfully modified resource one element in the returned array shall reference the instance of the CIM_ResourceAllocationSettingData class that represents the modified resource allocation or resource allocation request.
- 1114
- 1115
- 1116
- 1117 • If the method was executed synchronously and failed completely, the value shall be set to NULL.
- 1118

1119 **8.2.4.3 Return Codes**

1120 An implementation shall indicate the result of the method execution by using the return code values specified in Table 9.

1121

1122 **Table 9 – ModifyResourceSettings() Method: Return Code Values**

Value	Description
0	Method was successfully executed; all modification requests were successfully processed.
1	Method is not supported.
2	Method execution failed, but some modification requests may have been processed.
3	Method execution failed because a timeout condition occurred, but some modification requests may have been processed.
4	Method execution failed because invalid parameters were specified by the client; no modification requests were processed.
5	Method execution failed because the implementation does not support modifications on virtual resource allocations for the present virtual system state of the virtual system scoping virtual resources affected by this resource allocation modification request.
6	Method execution failed because incompatible parameters were specified by the client; no modification requests were processed.
4096	Method execution is performed asynchronously. The specifications given in 8.1.3 apply.
NOTE: Even if the return code indicates a failure, some modification requests may have been successfully executed. In this case, the set of successfully modified resources is conveyed through the value of the ResultingResourceSettings parameter.	

1123 **8.2.5 CIM_VirtualSystemManagementService.ModifySystemSettings() Method**
 1124 **(Conditional)**

1125 Behavior applicable to all extrinsic methods is specified in 8.1.2.

1126 The implementation of the ModifySystemSettings() method is conditional with respect to the implementation supporting the modification of virtual systems and resources (see 7.4.6.3).

1127

1128 The execution of the ModifySystemSettings() method shall effect the modification of system settings, such that non-key and non-NULL values of the instance of the CIM_VirtualSystemSettingData class that is provided through the SystemSettings parameter override respective values in the instance identified through the value of the InstanceID property.

1129

1130

1131

1132 Table 10 contains requirements for parameters of this method.

1133 **Table 10 – ModifySystemSettings() Method: Parameters**

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

1134 8.2.5.1 SystemSettings Parameter

1135 A client shall set the SystemSettings parameter. Any instance of the CIM_VirtualSystemSettingData class
 1136 that is passed in as a value of the SystemSettings parameter shall have a valid non-NULL value in the
 1137 InstanceID property that identifies a respective instance of the CIM_VirtualSystemSettingData class
 1138 existing within the implementation. A client shall obtain such an instance before invoking the
 1139 ModifySystemSettings() method (for example, by using an extrinsic method or intrinsic CIM operation
 1140 that yields a respective instance as a result). For example, the client may use the intrinsic GetInstance()
 1141 CIM operation. The client shall then modify the instance locally so that it reflects the desired modifications
 1142 and finally pass it back in as a value of the SystemSettings parameter.

1143 The implementation shall ignore any value of the SystemSettings parameter that does not identify,
 1144 through the value of the InstanceID key property, an existing instance of the
 1145 CIM_VirtualSystemSettingData class within the implementation.

1146 8.2.5.2 Return Codes

1147 An implementation shall indicate the result of the method execution by using the return code values speci-
 1148 fied in Table 11.

1149 **Table 11 – ModifySystemSettings() Method: Return Code Values**

Value	Description
0	Method was successfully executed.
1	Method is not supported.
2	Method execution failed.
3	Method execution failed because a timeout condition occurred.
4	Method execution failed because invalid parameters were specified by the client.
5	Method execution failed because the implementation does not support modifications on virtual system settings for the present virtual system state of the virtual system identified by the input system settings.
6	Method execution failed because incompatible parameters were specified by the client.
4096	Method execution is performed asynchronously. The specifications given in 8.1.3 apply.

1150 8.2.6 CIM_VirtualSystemManagementService.RemoveResourceSettings() Method 1151 (Conditional)

1152 Behavior applicable to all extrinsic methods is specified in 8.1.2.

1153 The implementation of the RemoveResourceSettings() method is conditional with respect to the support
 1154 of the addition and the removal of virtual resources to virtual systems (see 7.4.6.2).

1155 The execution of the RemoveResourceSettings() method shall effect the removal of resource allocation
 1156 requests identified by the value of elements of the ResourceSettings[] parameter.

1157 Table 12 contains requirements for parameters of this method.

1158 **Table 12 – RemoveResourceSettings() Method: Parameters**

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

1159 **8.2.6.1 ResourceSettings[] Array Parameter**

1160 A client shall set the ResourceSettings[] array parameter. The value of any element specified in the
 1161 ResourceSettings[] array parameter shall represent requests for the removal of virtual resource state
 1162 extensions, of virtual resource definitions, or both in the scope of one virtual system.

1163 **8.2.6.2 Return Codes**

1164 An implementation shall indicate the result of the method execution by using the return code values speci-
 1165 fied in Table 13.

1166 **Table 13 – RemoveResourceSettings() Method: Return Code Values**

Value	Description
0	Method execution was successful.
1	Method is not supported.
2	Method execution failed.
3	Method execution failed because a timeout condition occurred.
4	Method execution failed because invalid parameters were specified by the client.
4096	Method execution is performed asynchronously. The specifications given in 8.1.3 apply.

1167 **8.3 Methods of the CIM_VirtualSystemSnapshotService Class**

1168 This subclause models virtual system snapshot management in terms of methods of the
 1169 CIM_VirtualSystemSnapshotService class.

1170 **8.3.1 CIM_VirtualSystemSnapshotService.CreateSnapshot() Method (Conditional)**

1171 Behavior applicable to all extrinsic methods is specified in 8.1.2.

1172 The implementation of the CreateSnapshot() method is conditional with respect to the support of the
 1173 creation, destruction and application of virtual system snapshots (see 7.7.1.1).

1174 The execution of the CreateSnapshot() method shall effect the creation of a snapshot of the affected vir-
 1175 tual system. The snapshot shall have the type that is designated by the value of the SnapshotType
 1176 parameter (see 8.3.1.3).

1177 A full snapshot shall contain all information required to restore the complete virtual system and its re-
 1178 sources to exactly the situation that existed when the snapshot was created. Other types of snapshots
 1179 may contain less information.

1180 If the virtual system is in the “Active” virtual system state, it may continue to perform tasks but may be
 1181 temporarily paused as the creation of the snapshot requires the capturing of state information.

1182 Table 14 contains requirements for parameters of this method.

1183 **Table 14 – CreateSnapshot() Method: Parameters**

Qualifiers	Name	Type	Description/Values
IN	AffectedSystem	CIM_ComputerSystem REF	See 8.3.1.1.
IN	SnapshotSettings	string	See 8.3.1.2.
IN	SnapshotType	uint16	See 8.3.1.3.

Qualifiers	Name	Type	Description/Values
OUT	ResultingSnapshot	CIM_VirtualSystemSettingData REF	See 8.3.1.4.
OUT	Job	CIM_ConcreteJob REF	See 8.1.3.2.

1184 8.3.1.1 AffectedSystem Parameter

1185 A client shall set a value of the AffectedSystem parameter to refer to the instance of the
1186 CIM_ComputerSystem class that represents the virtual system that is the source for the snapshot.

1187 An implementation shall interpret the value of the AffectedSystem parameter to identify the virtual system
1188 that is the source for the snapshot.

1189 8.3.1.2 SnapshotSettings Parameter

1190 A client may set a value of the SnapshotSettings parameter with an embedded instance of a
1191 CIM_SettingData class. It is assumed that an implementation-specific class derived from
1192 CIM_SettingData contains additional implementation-specific properties that enable some control over
1193 characteristics of the snapshot process.

1194 An implementation shall use the value of the SnapshotSettings parameter to control the characteristics of
1195 the snapshot process.

1196 8.3.1.3 SnapshotType Parameter

1197 A client shall set the value of the SnapshotType parameter to designate the intended type of snapshot.
1198 The value shall be one of the values set in the SnapshotTypesSupported[] array property in the instance
1199 of the CIM_VirtualSystemSnapshotServiceCapabilities class that is related to the snapshot service.

1200 An implementation shall use the value of the SnapshotType parameter to determine the requested type of
1201 snapshot. If a value is not specified or is not one of the values set in the SnapshotTypesSupported[] array
1202 property in the instance of the CIM_VirtualSystemSnapshotServiceCapabilities class that is related to the
1203 snapshot service, an implementation shall fail the method execution and set a return code of 6 (Invalid
1204 Type).

1205 8.3.1.4 ResultingSnapshot Parameter

1206 The implementation shall set the value of the ResultingSnapshot parameter as follows:

- 1207 • If the method execution is performed synchronously and is successful, the value shall be set to
1208 reference the instance of the CIM_VirtualSystemSettingData class that represents the newly
1209 created virtual system snapshot.
- 1210 • If the method execution is performed synchronously and fails, or if the method execution is per-
1211 formed asynchronously, the value shall be set to NULL.
- 1212 • If the method execution is performed asynchronously and is successful, see 8.1.2 to locate the
1213 instance of the CIM_VirtualSystemSettingData class that represents the newly created virtual
1214 system snapshot.

1215 8.3.1.5 Return Codes

1216 An implementation shall indicate the result of the method execution by using the return code values speci-
1217 fied in Table 15.

1218 **Table 15 – CreateSnapshot() Method: Return Code Values**

Value	Description
-------	-------------

Value	Description
0	Method execution was successful.
1	Method is not supported.
2	Method execution failed.
3	Method execution failed because a timeout condition occurred.
4	Method execution failed because an invalid parameter was specified.
5	Method execution failed because the affected system is in a state in which the implementation rejects capturing a snapshot.
6	Method execution failed because no snapshot or an unsupported type of snapshot was requested.
4096	Method execution is performed asynchronously. The specifications given in 8.1.3 apply.

1219 **8.3.2 VirtualSystemSnapshotService.DestroySnapshot() Method (Conditional)**

1220 Behavior applicable to all extrinsic methods is specified in 8.1.2.

1221 The implementation of the DestroySnapshot() method is conditional with respect to the support of virtual
 1222 system snapshots (see 7.7.1.1).

1223 The execution of the DestroySnapshot() method shall effect the destruction of the affected virtual system
 1224 snapshot. Dependency relationships from other snapshots to the affected snapshot shall be updated so
 1225 that the affected snapshot is no longer referenced. If the snapshot was persistently established to be used
 1226 during virtual system activation, the implementation may assign a different snapshot to be used for subse-
 1227 quent virtual system activations, or may fall back to the “Default” virtual system configuration to be used
 1228 for future activations. If a virtual system was activated using the snapshot and is still in a state other than
 1229 the “Defined” virtual system state, the active virtual system shall not be affected by the execution of the
 1230 DestroySnapshot() method.

1231 Table 16 contains requirements for parameters of this method.

1232 **Table 16 – DestroySnapshot() Method: Parameters**

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

1233 **8.3.2.1 AffectedSnapshot Parameter**

1234 A client shall set a value of the AffectedSnapshot parameter to refer to the instance of the
 1235 CIM_VirtualSystemSettingData class that represents a snapshot.

1236 An implementation shall interpret the value of the AffectedSnapshot parameter to identify the snapshot
 1237 that is to be destroyed.

1238 **8.3.2.2 Return Codes**

1239 An implementation shall indicate the result of the method execution using the return code values specified
 1240 by Table 17.

1241

Table 17 – DestroySnapshot() Method: Return Code Values

Value	Description
0	Method execution was successful.
1	Method is not supported.
2	Method execution failed.
3	Method execution failed because a timeout condition occurred.
4	Method execution failed because an invalid parameter was specified.
5	Method execution failed because the affected snapshot is in a state in which the implementation rejects destroying a snapshot.
6	Method execution failed because the affected snapshot is of a type that is not destroyable.
4096	Method execution is performed asynchronously. The specifications given in 8.1.3 apply.

1242 8.3.3 VirtualSystemSnapshotService.ApplySnapshot() Method (Conditional)

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

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

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

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

1251 Table 18 contains requirements for parameters of this method.

1252

Table 18 – ApplySnapshot() Method: Parameters

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

1253 8.3.3.1 Snapshot Parameter

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

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

1258 8.3.3.2 Return Codes

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

1261

Table 19 – ApplySnapshot() Method: Return Code Values

Value	Description
-------	-------------

Value	Description
0	Method execution was successful.
1	Method is not supported.
2	Method execution failed.
3	Method execution failed because a timeout condition occurred.
4	Method execution failed because an invalid parameter was specified.
5	Method execution failed because the affected system is in a state where snapshots cannot be applied.
6	Method execution failed because the type of the affected system does not support the application of a snapshot.
4096	Method execution is performed asynchronously. The specifications given in 8.1.3 apply.

1262 **8.4 Profile Conventions for Operations**

1263 Support for operations for each profile class (including associations) is specified in the following sub-
 1264 clauses. Each subclause includes either the statement “All operations in the default list in 8.4 are
 1265 supported as described by DSP0200 version 1.2.0 or a table listing all of the operations that are not sup-
 1266 ported by this profile or where the profile requires behavior other than that described by DSP0200 version
 1267 1.2.0.

1268 The default list of operations is as follows:

- 1269 • GetInstance
- 1270 • Associators
- 1271 • AssociatorNames
- 1272 • References
- 1273 • ReferenceNames
- 1274 • EnumerateInstances
- 1275 • EnumerateInstanceNames

1276 A compliant implementation shall support all of the operations in the default list for each class, unless the
 1277 “Requirement” column states something other than *Mandatory*.

1278 This profile specification defines methods in terms of DSP0200 version 1.2.0.

1279 **8.4.1 CIM_AffectedJobElement**

1280 Table 20 lists operations that either have special requirements beyond those from DSP0200 version 1.2.0
 1281 or shall not be supported.

1282 **Table 20 – Operations: CIM_AffectedJobElement Association**

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

EnumerateInstances	Unspecified	None
EnumerateInstanceNames	Unspecified	None

1283 **8.4.2 CIM_ComputerSystem**

1284 All operations in the default list in 8.4 are supported as described by DSP0200 version 1.2.0.

1285 **8.4.3 CIM_ConcreteJob**

1286 All operations in the default list in 8.4 are supported as described by DSP0200 version 1.2.0.

1287 **8.4.4 CIM_Dependency**

1288 Table 22 lists operations that either have special requirements beyond those from DSP0200 version 1.2.0
1289 or shall not be supported.

1290 **Table 21 – Operations: CIM_Dependency Association**

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

1291 **8.4.5 CIM_ElementCapabilities**

1292 Table 22 lists operations that either have special requirements beyond those from DSP0200 version 1.2.0
1293 or shall not be supported.

1294 **Table 22 – Operations: CIM_ElementCapabilities Association**

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

1295 **8.4.6 CIM_ElementConformsToProfile**

1296 Table 23 lists operations that either have special requirements beyond those from DSP0200 version 1.2.0
1297 or shall not be supported.

1298

Table 23 – Operations: CIM_ElementConformsToProfile Association

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

1299

8.4.7 CIM_HostedDependency

1300

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

1301

1302

Table 24 – Operations: CIM_HostedDependency Association

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

1303

8.4.8 CIM_HostedService

1304

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

1305

1306

Table 25 – Operations: CIM_HostedService Association

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

1307

8.4.9 CIM_LastAppliedSnapshot

1308

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

1309

1310

Table 26 – Operations: CIM_LastAppliedSnapshot Association

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

1311 **8.4.10 CIM_MostCurrentSnapshotInBranch**

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

1314

Table 27 – Operations: CIM_MostCurrentSnapshotInBranch Association

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

1315 **8.4.11 CIM_ReferencedProfile**

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

1318

Table 28 – Operations: CIM_ReferencedProfile Association

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

1319 **8.4.12 CIM_RegisteredProfile**

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

1321 **8.4.13 CIM_ServiceAffectsElement**

1322 Table 29 lists operations that either have special requirements beyond those from DSP0200 version 1.2.0
 1323 or shall not be supported.

1324 **Table 29 – Operations: CIM_ServiceAffectsElement Association**

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

1325 **8.4.14 CIM_SnapshotOfVirtualSystem**

1326 Table 30 lists operations that either have special requirements beyond those from DSP0200 version 1.2.0
 1327 or shall not be supported.

1328 **Table 30 – Operations: CIM_SnapshotOfVirtualSystem Association**

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

1329 **8.4.15 CIM_System**

1330 All operations in the default list in 8.4 are supported as described by DSP0200 version 1.2.0.

1331 **8.4.16 CIM_VirtualSystemManagementCapabilities**

1332 All operations in the default list in 8.4 are supported as described by DSP0200 version 1.2.0.

1333 **8.4.17 CIM_VirtualSystemManagementService**

1334 All operations in the default list in 8.4 are supported as described by DSP0200 version 1.2.0.

1335 **8.4.18 CIM_VirtualSystemSnapshotService**

1336 All operations in the default list in 8.4 are supported as described by DSP0200 version 1.2.0.

1337 **8.4.19 CIM_VirtualSystemSnapshotCapabilities**

1338 All operations in the default list in 8.4 are supported as described by DSP0200 version 1.2.0.

1339 **8.4.20 CIM_VirtualSystemSnapshotServiceCapabilities**

1340 All operations in the default list in 8.4 are supported as described by DSP0200 version 1.2.0.

1341 **9 Use Cases (Informative)**

1342 The following use cases and object diagrams illustrate use of –this profile. They are for informational pur-
1343 poses only and do not introduce behavioral requirements for implementations of the profile.

1344 **9.1 General Assumptions**

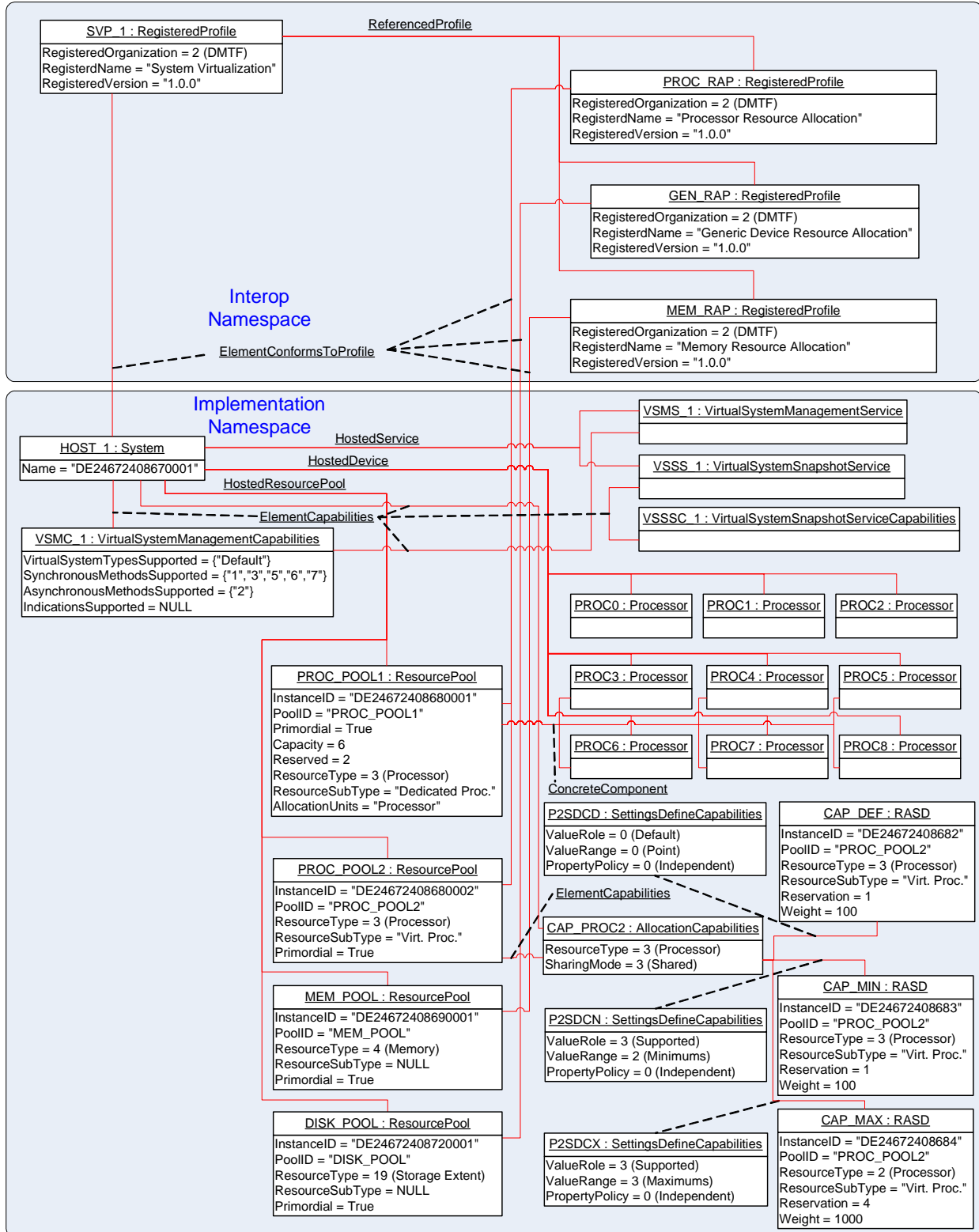
1345 For all use cases, it is assumed that a client performs intrinsic CIM operations, extrinsic CIM operations,
1346 or both.

1347 For all use cases except the use case described in 9.2.1, the following conditions are implicitly assumed:

- 1348 • The client knows the URL of a CIM object manager (CIMOM) that exposes an implementation of
1349 this profile.
- 1350 • The client is able to communicate with the CIMOM through a specified CIM protocol. An exam-
1351 ple is the use of the http protocol as described in DSP0200. The client may use a facility like a
1352 CIM client API to perform the encoding and decoding of CIM messages.

1353 **9.2 Discovery, Localization, and Inspection**

1354 This set of use cases describes how a client obtains access to an implementation, detects the central and
1355 scoped instances, and analyzes information available through these instances. Figure 3 outlines a sample
1356 situation that is referenced by some of the use-case descriptions in subsequent subclauses.



1357

1358 **Figure 3 – System Virtualization Profile Instance Diagram: Discovery, Localization, and Inspection**

1359 **9.2.1 SLP-Based Discovery of CIM Object Managers Hosting Implementations of This** 1360 **Profile**

1361 The service location protocol (SLP) is used to locate CIM object managers (CIMOMs). A CIMOM that
1362 implements SLP as a discovery mechanism is required to register with SLP all instances of the
1363 CIM_RegisteredProfile class that reside in the Interop namespace. An SLP service type is used to identify
1364 entities that are registered with SLP. An SLP service type is a structured string variable.

1365 **Assumption:** This profile is registered by at least one CIMOM that maintains a registration with an SLP
1366 Directory Agent. The registration includes information about registered DMTF management profiles. The
1367 client is able to make SLP calls.

- 1368 • The client invokes the SLPFindSrvs() SLP function as follows:
 - 1369 – The value of the srvtype parameter is set to “service:wbem”.
 - 1370 – The value of the scopelist parameter is set to “default”.
 - 1371 – The value of the filter parameter is set to “(RegisteredProfilesSupported=DMTF:System
1372 Virtualization)”.

1373 **Result:** Each URL in a list of URLs identifies a CIMOM where this profile is implemented.

1374 **9.2.2 Locate Conformant Implementations Using the EnumerateInstances() Operation**

1375 **Assumption:** The client knows the URL of a CIMOM hosting implementations of this profile (see 9.2.1).

- 1376 1) Using the URL, the client invokes the intrinsic EnumerateInstances() CIM operation with the
1377 value of the ClassName input parameter set to “CIM_RegisteredProfile”.
1378 The result is a list of instances of the CIM_RegisteredProfile class.
- 1379 2) The client iterates over the list of instances of the CIM_RegisteredProfile class and selects in-
1380 stances where
 - 1381 – the RegisteredOrganization property has a value of 2 (DMTF)
 - 1382 – the RegisteredName property has a value of “System Virtualization”
 - 1383 – the RegisteredVersion property has a value equal to or greater than “1.0.0”

1384 **Result:** The client knows a set of instances of the CIM_RegisteredProfile class, each representing an im-
1385 plementation of this profile.

1386 In the example shown in Figure 3, one instance of the CIM_RegisteredProfile class represents an imple-
1387 mentation of this profile; it is tagged SVP_1.

1388 **9.2.3 Locate Conformant Implementations Using the ExecuteQuery() Operation**

1389 **Assumption:** The client knows the URL of a CIMOM hosting implementations of this profile (see 9.2.1).

- 1390 • Using the URL, the client invokes the intrinsic ExecuteQuery() CIM operation as follows:
 - 1391 – The value of the QueryLanguage input parameter is set to “CIM:CQL”.
 - 1392 – The value of the Query input parameter is set to “SELECT * FROM CIM_RegisteredProfile
1393 WHERE RegisteredName = ‘System Virtualization’ AND RegisteredVersion >= ‘1.0.0’”.

1394 **Result:** The client knows a set of instances of the CIM_RegisteredProfile class, each representing an im-
1395 plementation of this profile.

1396 In the example shown in Figure 3, one instance of the CIM_RegisteredProfile class represents an imple-
1397 mentation of this profile; it is tagged SVP_1.

1398 9.2.4 Locate Host Systems That Are Central and Scoping Instances of This Profile

1399 **Assumption:** The client knows a reference to an instance of the CIM_RegisteredProfile class that
1400 represents an implementation of this profile (see 9.2.2 or 9.2.3).

- 1401 • The client invokes the intrinsic AssociatorNames() CIM operation as follows:
 - 1402 – The value of the ObjectName parameter is set to refer to the instance of the
 - 1403 CIM_RegisteredProfile class.
 - 1404 – The value of the AssocClass parameter is set to “CIM_ElementConformsToProfile”.
 - 1405 – The value of the ResultClass parameter is set to “CIM_System”.

1406 **Result:** The client knows a set of references to instances of the CIM_System class that represent host
1407 systems that are central and scoping instances of this profile.

1408 In the example shown in Figure 3, one instance of the CIM_RegisteredProfile class represents a host sys-
1409 tem that is a central and scoping instance of this profile; it is tagged HOST_1.

1410 9.2.5 Locate Implementations of Scoped Resource Allocation DMTF Management 1411 Profiles

1412 **Assumption:** The client knows a reference to an instance of the CIM_RegisteredProfile class that
1413 represents an implementation of this profile (see 9.2.2 or 9.2.3).

- 1414 1) The client invokes the intrinsic Associators() CIM operation to obtain a the list of scoped DMTF
1415 management profiles, as follows:
 - 1416 – The value of the ObjectName parameter is set to refer to the instance of the
 - 1417 CIM_RegisteredProfile class.
 - 1418 – The value of the AssocClass parameter is set to “CIM_ReferencedProfile”.
 - 1419 – The value of the ResultClass parameter is set to “CIM_RegisteredProfile”.

1420 The result is a set of instances of the CIM_RegisteredProfile class that each represent an imple-
1421 mentation of a DMTF management profile that is scoped by the *System Virtualization Profile*.
- 1422 2) For each instance of the CIM_RegisteredProfile class, the client determines whether the value
1423 of the RegisteredName property matches the registered name of one of the scoped resource
1424 allocation DMTF management profiles as specified by Table 1.

1425 If the value does not match any name of a resource allocation DMTF management profile
1426 scoped by this profile, the client ignores that instance of the CIM_RegisteredProfile class.

1427 **Result:** The client knows a set of instances of the CIM_RegisteredProfile class that each represent an im-
1428 plementation of a resource allocation DMTF management profile that is scoped by the *System*
1429 *Virtualization Profile*.

1430 In the example shown in Figure 3, three instances of the CIM_RegisteredProfile class are associated with
1431 the instance of the CIM_RegisteredProfile class that is tagged SVP_1 and represents a central instance
1432 of the *System Virtualization Profile*. These instances represent implementations of scoped resource
1433 allocation DMTF management profiles:

- 1434 • The instance tagged PROC_RAP represents an implementation of the *Processor Device*
1435 *Resource Virtualization Profile*.
- 1436 • The instance tagged GEN_RAP represents an implementation of the *Generic Device Resource*
1437 *Virtualization Profile*.
- 1438 • The instance tagged MEM_RAP represents an implementation of the *Memory Resource*
1439 *Virtualization Profile*.

1440 9.2.6 Locate Virtual System Management Service

1441 **Assumption:** The client knows a reference to an instance of the CIM_System class that represents a
1442 host system that is a central instance of this profile (see 9.2.4).

- 1443
- The client invokes the intrinsic AssociatorNames() CIM operation as follows:
 - 1444 – The value of the ObjectName parameter is set to refer to the instance of the CIM_System
 - 1445 class.
 - 1446 – The value of the AssocClass parameter is set to “CIM_HostedService”.
 - 1447 – The value of the ResultClass parameter is set to “CIM_VirtualSystemManagementService”.

1448 **Result:** The client knows a reference to the instance of the CIM_VirtualSystemManagementService class
1449 that represents the virtual system management service that serves the host system. If the operation is
1450 successful, the size of the result set is 1.

1451 In the example shown in Figure 3, one instance of the CIM_VirtualSystemManagementService class
1452 serves the host system; it is tagged VSMS_1.

1453 9.2.7 Determine the Capabilities of an Implementation

1454 **Assumption:** The client knows a reference to an instance of the CIM_System class that represents a
1455 host system that is a central instance of this profile (see 9.2.4).

- 1456
- 1) The client invokes the intrinsic Associators() CIM operation as follows:
 - 1457 – The value of the ObjectName parameter is set to refer to the instance of the CIM_System
 - 1458 class.
 - 1459 – The value of the AssocClass parameter is set to “CIM_ElementCapabilities”.
 - 1460 – The value of the ResultClass parameter is set to
 - 1461 “CIM_VirtualSystemManagementCapabilities”.

1462 The result is a list of instances of the CIM_VirtualSystemManagementCapabilities class. If the
1463 operation is successful, the size of the result set is 1.
 - 2) The client analyzes the instance of the CIM_VirtualSystemManagementCapabilities class.
 - 1465 – The VirtualSystemTypesSupported[] array property lists identifiers of virtual system types
 - 1466 that the implementation supports.
 - 1467 – The SynchronousMethodsSupported[] array property lists identifiers of methods of the
 - 1468 CIM_VirtualSystemManagementService class that are supported with synchronous method
 - 1469 execution only.
 - 1470 – The AsynchronousMethodsSupported[] array property lists identifiers of methods of the
 - 1471 CIM_VirtualSystemManagementService class that are supported with synchronous and
 - 1472 asynchronous method execution.
 - 1473 – The IndicationsSupported[] array property lists identifiers of types of indications that the
 - 1474 implementation supports.

1475 **Result:** The client knows the capabilities of the host system in terms of properties of the
1476 CIM_VirtualSystemManagementCapabilities class.

1477 In the example shown in Figure 3, one instance of the CIM_VirtualSystemManagementCapabilities class
1478 is associated with the host system; it is tagged VSMC_1.

- 1479
- The VirtualSystemTypesSupported[] array property lists one element with the value “Default”,
1480 which indicates that the implementation supports one virtual system type named “Default”. The
1481 semantics are implementation specific.

- 1482 • The SynchronousMethodsSupported[] array property lists enumerated values:
1483 { 1 (AddResourceSettingsSupported), 3 (DestroySystemSupported),
1484 5 (ModifyResourceSettingsSupported), 6 (ModifySystemSettingsSupported), and
1485 7 (RemoveResourcesSupported) }, which indicates that the AddResources() method, the
1486 DestroySystem() method, the ModifyResourceSettings() method, and the
1487 RemoveResourceSettings() method are supported by the implementation with synchronous
1488 execution.
- 1489 • The AsynchronousMethodsSupported[] array property lists the enumerated value
1490 { 2 (DefineSystemSupported) }, which indicates that the DefineSystem() method is supported
1491 by the implementation with synchronous or asynchronous execution.
- 1492 • The value of the IndicationsSupported[] array property is NULL, which indicates that indications
1493 are not supported by the implementation.

1494 9.2.8 Locate Hosted Resource Pools of a Particular Resource Type

1495 **Assumption:** The client knows a reference to an instance of the CIM_System class that represents a
1496 host system that is a central instance of this profile (see 9.2.4).

- 1497 1) The client invokes the intrinsic Associators() CIM operation as follows:
 - 1498 – The value of the ObjectName parameter is set to refer to the instance of the CIM_System
1499 class.
 - 1500 – The value of the AssocClass parameter is set to “CIM_HostedResourcePool”.
 - 1501 – The value of the ResultClass parameter is set to “CIM_ResourcePool”.
- 1502 The result is a list of instances of the CIM_ResourcePool class.
- 1503 2) For each instance of CIM_ResourcePool, the client determines whether the value of the
1504 ResourceType property matches the requested resource type.
- 1505 If the value does not match the requested resource type, the client drops that instance of the
1506 CIM_ResourcePool class from the list.

1507 **Result:** The client knows a set of instances of the CIM_ResourcePool class, each representing a hosted
1508 resource pool of the requested resource type.

1509 9.2.9 Obtain a Set of Central Instances of Scoped Resource Allocation DMTF 1510 Management Profiles

1511 Resource allocation DMTF management profiles are based on the abstract *Resource Allocation Profile*
1512 that defines the CIM_ResourcePool class as the central class. The procedure for the determination of
1513 central instances of scoped DMTF management profiles depends on the profile advertisement
1514 methodology applied by the respective implementations.

1515 **Assumption:** The client knows a reference to an instance of the CIM_RegisteredProfile class that
1516 represents an implementation of a scoped DMTF management profile (see 9.2.5).

- 1517 • The client invokes the intrinsic Associators() CIM operation to obtain the list of instances of the
1518 CIM_ResourcePool class that are central instances of the scoped DMTF management profiles,
1519 as follows:
 - 1520 – The value of the ObjectName parameter is set to refer to the instance of the
1521 CIM_RegisteredProfile class
 - 1522 – The value of the AssocClass parameter is set to “CIM_ElementConformsToProfile”.
 - 1523 – The value of the ResultClass parameter is set to “CIM_ResourcePool”.
- 1524 The result is a list of instances of the CIM_ResourcePool class; the list may be empty.

- 1525 – If the list is not empty, the *central class* profile implementation advertisement methodology
1526 is applied by the implementation for the scoped resource allocation DMTF management
1527 profile. In this case, the list is the result for this use case.
- 1528 – If the list is empty, the *scoping class* profile implementation advertisement methodology is
1529 applied by the implementation for the scoped resource allocation DMTF management pro-
1530 file. In this case, the client
- 1531 – needs to know the resource type associated with the scoped resource allocation
1532 DMTF management profile
- 1533 – applies use case 9.2.8 to obtain a list of instances of the CIM_ResourcePool class
1534 that each represent a resource pool of that particular resource type.
- 1535 The resulting list is the result for this use case.

1536 **Result:** The client knows a list of instances of the CIM_ResourcePool class, each representing a central
1537 instance of a scoped resource allocation DMTF management profile.

1538 9.2.10 Determine the Supported Resource Types of an Implementation

1539 **Assumption:** The client knows a reference to an instance of the CIM_RegisteredProfile class that
1540 represents an implementation of this profile (see 9.2.2 or 9.2.3).

- 1541 1) The client locates implementations of DMTF management profiles that are scoped by this profile
1542 (see 9.2.5).
- 1543 The result is a list of references to instances of the CIM_RegisteredProfile class that represent
1544 implementations of DMTF management profiles that are scoped by this profile.
- 1545 2) For each instance of CIM_RegisteredProfile, the client obtains the set of instances of the
1546 CIM_ResourcePool class that are central instances of the respective scoped resource allocation
1547 DMTF management profiles and represent a conformant resource pool (see 9.2.9).
- 1548 The result is a list of instances of the CIM_ResourcePool class that are central instances of
1549 scoped resource allocation DMTF management profiles.
- 1550 3) The client creates an initially empty list of integer values. For each instance that is a result from
1551 step 2), the client determines whether the value of property ResourceType is already repre-
1552 sented in the list:
- 1553 – If that value is already contained in the list, the client ignores the element.
- 1554 – If that value is not yet contained in the list, the client adds a new element to the list with
1555 that value.

1556 **Result:** The client knows a list of integer values, each designating a resource type that is supported by
1557 the implementation.

1558 In the example shown in Figure 3, three instances of the CIM_RegisteredProfile class are associated with
1559 the instance of the CIM_RegisteredProfile class that represents the implementation of this profile. These
1560 instances are central instances of scoped resource allocation DMTF management profiles:

- 1561 • The instance tagged PROC_RAP represents an implementation of the *Processor Device*
1562 *Resource Virtualization Profile*.
- 1563 • The instance tagged GEN_RAP represents an implementation of the *Generic Device Resource*
1564 *Virtualization Profile*.
- 1565 • The instance tagged MEM_RAP represents an implementation of the *Memory Resource*
1566 *Virtualization Profile*.

1567 These instances are all associated with respective instances of the CIM_ResourcePool class, indicating
1568 that in this example in all cases the central class profile advertisement methodology is in use:

- 1569 • The instance tagged PROC_RAP is associated with two instances that represent resource
1570 pools for the allocation of processors. They show a value of 3 (Processor) for the ResourceType
1571 property and are tagged PROC_POOL1 and PROC_POOL2.
- 1572 • The instance tagged GEN_RAP is associated with one instance that represents a resource pool
1573 for the allocation of virtual disks. It shows a value of 19 (Storage Extent) for the ResourceType
1574 property and is tagged DISK_POOL.
- 1575 • The instance tagged MEM_RAP is associated with one instance that represents a resource pool
1576 for the allocation of memory. It shows a value of 4 (Memory) for the ResourceType property and
1577 is tagged MEM_POOL.

1578 The resulting list of integer values is {"3","4","19"} and designates the implemented resource types
1579 3 (Processor), 4 (Memory), and 19 (Storage Extent).

1580 9.2.11 Determine the Default Resource Pool for a Resource Type

1581 **Assumption:** The client knows a reference to an instance of the CIM_System class that represents a
1582 host system that is a central instance of this profile (see 9.2.4).

- 1583 1) The client invokes the intrinsic Associators() CIM operation for a list of allocation capabilities
1584 associated with resource pools hosted by the host system, as follows:
 - 1585 – The value of the ObjectName parameter is set to refer to the instance of the CIM_System
1586 class.
 - 1587 – The value of the AssocClass parameter is set to "CIM_ElementCapabilities".
 - 1588 – The value of the ResultClass parameter is set to "CIM_AllocationCapabilities".

1589 The result is a list of instances of the CIM_AllocationCapabilities class.
- 1590 2) The client drops instances from the result list of step 1) that have a value for the ResourceType
1591 property that does not match the requested resource type.

1592 The purpose of the following two steps is to further limit the result set from step 2) to those in-
1593 stances of the CIM_AllocationCapabilities class that describe default settings. Default settings
1594 are flagged in the connecting instance of the CIM_ElementCapabilities association that has a
1595 value of 2 (Default) for the Characteristics property.
- 1596 3) For each instance of the list resulting from step 2), the client invokes the intrinsic References()
1597 CIM operation for a list of association instances that refer to the resource pool:
 - 1598 – The value of the ObjectName parameter refers the instance of the CIM_ResourcePool
1599 class.
 - 1600 – The value of the ResultClass parameter is set to "CIM_AllocationCapabilities".

1601 The result is a list of instances of the CIM_ElementCapabilities association that associate an in-
1602 stance of the CIM_ResourcePool class that is taken from the result of step 2).
- 1603 4) From the list obtained in step 3), the client drops all elements that meet either of the following
1604 conditions:
 - 1605 – have a value other than 2 (Default) for the Characteristics property
 - 1606 – do not refer to the instance of the CIM_System class that represents the host system
1607 through the ManagedElement property

1608 The list should now contain one instance of the CIM_AllocationCapabilities class that represents
1609 default allocation capabilities for the resource type in question.

1610 5) The client invokes the intrinsic Associators() CIM operation to resolve association for the re-
1611 source pool, as follows:

- 1612 – The value of the ObjectName parameter refers to the instance of the
1613 CIM_AllocationCapabilities class selected in step 4).
- 1614 – The value of the AssocClass parameter is set to “CIM_ElementCapabilities”.
- 1615 – The value of the ResultClass parameter is set to “CIM_ResourcePool”.

1616 The result is a list of instances of the CIM_ResourcePool class. The size of the list is 1.

1617 **Result:** The client knows the instance of the CIM_ResourcePool class that represents the default re-
1618 source pool for the requested resource type.

1619 In the example shown in Figure 3, allocation capabilities are depicted only for the virtual processor pool.
1620 In the subsequent description, it is assumed that the client looks for the default resource pool for proces-
1621 sors:

- 1622 • With step 1) of this use case, the client resolves the CIM_ElementCapabilities association from
1623 the instance of the CIM_System class that represents the host system (tagged HOST_1) to in-
1624 stances of the CIM_AllocationCapabilities class. A conformant implementation of the *Allocation*
1625 *Capabilities Profile* shows only one associated element for each resource type.
- 1626 • With step 2), the client reduces the result set to the one element that describes allocation
1627 capabilities processors. This instance is tagged CAP_PROC1.
- 1628 • With steps 3) and 4), the client further reduces the result set to the one instance of the
1629 CIM_AllocationCapabilities class that represents the system’s default capabilities for resource
1630 type 3 (Processor).
- 1631 • With step 5), the client resolves the CIM_ElementCapabilities association in order to obtain the
1632 instance of the CIM_ResourcePool class that represents the default resource pool for
1633 processors. This instance is tagged PROC_POOL2.

1634 **9.2.12 Determine Resource Pool for a Resource Allocation Request or Allocated** 1635 **Resource**

1636 **Assumption:** The client knows a reference to an instance of the CIM_ResourceAllocationSettingData
1637 class that represents a resource allocation request or allocated resource.

- 1638 • The client invokes the intrinsic Associators() CIM operation for a list of allocation capabilities
1639 associated with resource pools hosted by the host system, as follows:
 - 1640 – The value of the ObjectName parameter is set to refer to the instance of the
1641 CIM_ResourceAllocationSettingData class.
 - 1642 – The value of the AssocClass parameter is set to “CIM_ResourceAllocationFromPool”.
 - 1643 – The value of the ResultClass parameter is set to “CIM_ResourcePool”.

1644 The result is a list of instances of the CIM_ResourcePool class containing one element.

1645 **Result:** The client knows the instance of the CIM_ResourcePool class that represents the resource pool
1646 for the resource allocation request or allocated resource.

1647 **9.2.13 Determine Valid Settings for a Resource Type**

1648 This use case describes the determination of valid settings for a resource type in the context of either the
1649 system as a whole or one resource pool.

1650 **Assumption:** The client knows a reference to either of the following instances:

- 1651 • an instance of the CIM_ResourcePool class that represents a resource pool that is a central in-
1652 stance of a resource allocation DMTF management profile
 - 1653 • an instance of the CIM_System class that represents a host system
- 1654 The sequence of activities is as follows:
- 1655 1) The client invokes the intrinsic Associators() CIM operation as follows:
 - 1656 – The value of the ObjectName parameter is set to refer to the instance of the
1657 CIM_ResourcePool class or the CIM_System class.
 - 1658 – The value of the AssocClass parameter is set to “CIM_ElementCapabilities”.
 - 1659 – The value of the ResultClass parameter is set to “CIM_AllocationCapabilities”.

1660 The result is a list of instances of the CIM_AllocationCapabilities class that describe the
1661 capabilities of the input instance.
 - 1662 2) The client drops from the result of step 1) those instances in which the ResourceType property
1663 designates a resource type other than the requested resource type. This step is required only if
1664 the starting point of the use case was an instance of the CIM_System class.

1665 At this point the client has a list of instances of the CIM_AllocationCapabilities class that de-
1666 scribe allocation capabilities. The value of the SharingMode property allows a distinction
1667 between shared and dedicated resources.
 - 1668 3) The client invokes the intrinsic References() CIM operation for a set of instances of the
1669 CIM_SettingsDefineCapabilities association that each associate one instance of the
1670 CIM_ResourceAllocationSettingData class that describes a limiting aspect (min/max/increment),
1671 as follows:
 - 1672 – The value of the ObjectName parameter is set to refer to the instance of the
1673 CIM_AllocationCapabilities class.
 - 1674 – The value of the ResultClass parameter is set to “CIM_SettingsDefineCapabilities”.

1675 The result is a list of instances of the CIM_SettingsDefineCapabilities association.
 - 1676 4) For each instance that is a result from step 3), the client analyzes the values of the
1677 PropertyPolicy property and the ValueRange property. The value of the ValueRole property is
1678 irrelevant in this case.

1679 The property values have the following impact:

 - 1680 – The value of the PropertyPolicy property is 0 (Independent) for a conformant
1681 implementation of the *Allocation Capabilities Profile* in association instances that connect a
1682 min/max/increment limiting setting.
 - 1683 – The value of the ValueRange property allows determining the designation of the associated
1684 setting:
 - 1685 – A value of 1 (Minimums) indicates that the referenced instance of the
1686 CIM_ResourceAllocationSettingData class represents a lower limit for the allocation of
1687 resources of the respective resource type.
 - 1688 – A value of 2 (Maximums) indicates that the referenced instance of the
1689 CIM_ResourceAllocationSettingData class represents an upper limit for the allocation
1690 of resources of the respective resource type.
 - 1691 – A value of 3 (Increments) indicates that the referenced instance of the
1692 CIM_ResourceAllocationSettingData class represents an increment for the allocation
1693 of resources of the respective resource type.
 - 1694 5) For each association instance obtained in step 4), the client invokes the intrinsic GetInstance()
1695 CIM operation for the instance of the CIM_ResourceAllocationSettingData class that describes

1696 the respective limitation. The value of InstanceName parameter is set to the value of the
1697 PartComponent property in the association instance obtained in step 4).

1698 In each case, the result is an instance of the CIM_ResourceAllocationSettingData class that
1699 represents a limiting setting.

1700 **Result:** The client knows the valid resource settings for the requested resource type.

1701 9.2.14 Determine Implementation Class Specifics

1702 The *System Virtualization Profile* specifies the use of classes derived from the CIM_SettingData class,
1703 namely the CIM_VirtualSystemSettingData class and the CIM_ResourceAllocationSettingData class.
1704 Instances of these classes are used to describe requirements on virtual systems and virtual resources as
1705 these are created or modified. An implementation may provide platform-specific implementation classes
1706 that extend these classes (or, for the CIM_ResourceAllocationSettingData class, that extend resource-
1707 type-specific extensions specified in a resource-type-specific resource allocation DMTF management pro-
1708 file).

1709 A client should be prepared to deal with these extensions. A client should obtain class information for all
1710 derived classes it deals with, in particular focusing on all class qualifiers and all property qualifiers,
1711 namely

- 1712 • the Description qualifier that provides a description of the subclass or property
- 1713 • the DisplayName qualifier that provides a name for each subclass or property that is potentially
1714 known to end-users

1715 **Assumption:** The client knows a reference to an instance of the class for which the client wants to obtain
1716 class-specific information.

- 1717 1) The client extracts the class name from the reference.
- 1718 2) The client invokes the intrinsic GetClass() CIM operation to obtain a formal class description,
1719 as follows:
 - 1720 – The value of the ClassName parameter is set to the name of the class.
 - 1721 – The value of the LocalOnly parameter is set to “false”.
 - 1722 – The value of the IncludeQualifiers parameter is set to “true”.
 - 1723 – The value of the IncludeClassOrigin parameter is set to “true”.

1724 The result is a description of a CIM class.

1725 **Result:** The client has a description of the class. The format depends on the CIM client used to issue the
1726 request and is based on the XML class data structure that describes a CIM class as defined in the
1727 *Specification for the Representation of CIM in XML*, version 2.2.0. The description contains the class's
1728 qualifiers, its properties with property qualifiers, and its methods with method qualifiers. Inspection of the
1729 class description enables the client to create local instances of the respective implementation class.

1730 9.2.15 Determine the Implementation Class for a Resource Type

1731 **Assumption:** The client knows a list of references to instances of the CIM_ResourcePool class that
1732 represent resource pools available at a host system.

- 1733 1) The client applies use case 9.2.13 to obtain a reference to an instance of the
1734 CIM_ResourceAllocationSettingData class that is associated with an instance of the
1735 CIM_ResourcePool class of the requested type through an instance of the
1736 CIM_SettingsDefineCapabilities association with the ValueRole property set to “DEFAULT”.
- 1737 2) The client applies use case 9.2.14 to obtain class information about that instance.

1738 **Result:** The client has an implementation class descriptor, which allows the client to analyze the
 1739 implementation class for its qualifiers, its properties and their qualifiers, and its methods and their
 1740 qualifiers. Further, the client can create local instances of the returned class that may be used as input on
 1741 methods of the CIM_VirtualSystemManagementService class.

1742 9.2.16 Locate Virtual Systems Hosted by a Host System

1743 **Assumption:** The client knows a reference to an instance of the CIM_System class that is the central in-
 1744 stance of this profile and represents a host system (see 9.2.4).

- 1745 • The client invokes the intrinsic AssociatorNames() CIM operation for the list of virtual systems,
 1746 as follows:
 - 1747 – The value of the ObjectName parameter is set to refer to the instance of the CIM_System
 1748 class.
 - 1749 – The value of the AssocClass parameter is set to “CIM_HostedSystem”.
 - 1750 – The value of the ResultClass parameter is set to “CIM_ComputerSystem”.
- 1751 The result is a list of references to instances of the CIM_ComputerSystem class.

1752 **Result:** The client knows a set of references to instances of the CIM_ComputerSystem class that
 1753 represent virtual systems that are hosted by the host system.

1754 9.3 Virtual System Definition, Modification, and Destruction

1755 **General assumption:** The client knows a reference to an instance of the
 1756 CIM_VirtualSystemManagementService class that represents the virtual system management services of
 1757 a host system (see 9.2.6).

1758 9.3.1 Virtual System Definition

1759 Virtual system definition is performed using a client-provided configuration, a configuration of an existing
 1760 virtual system, a configuration that is stored within the implementation, or combinations of these.

1761 9.3.1.1 Define Virtual System Based on "Input" and "Reference" Virtual System Configuration

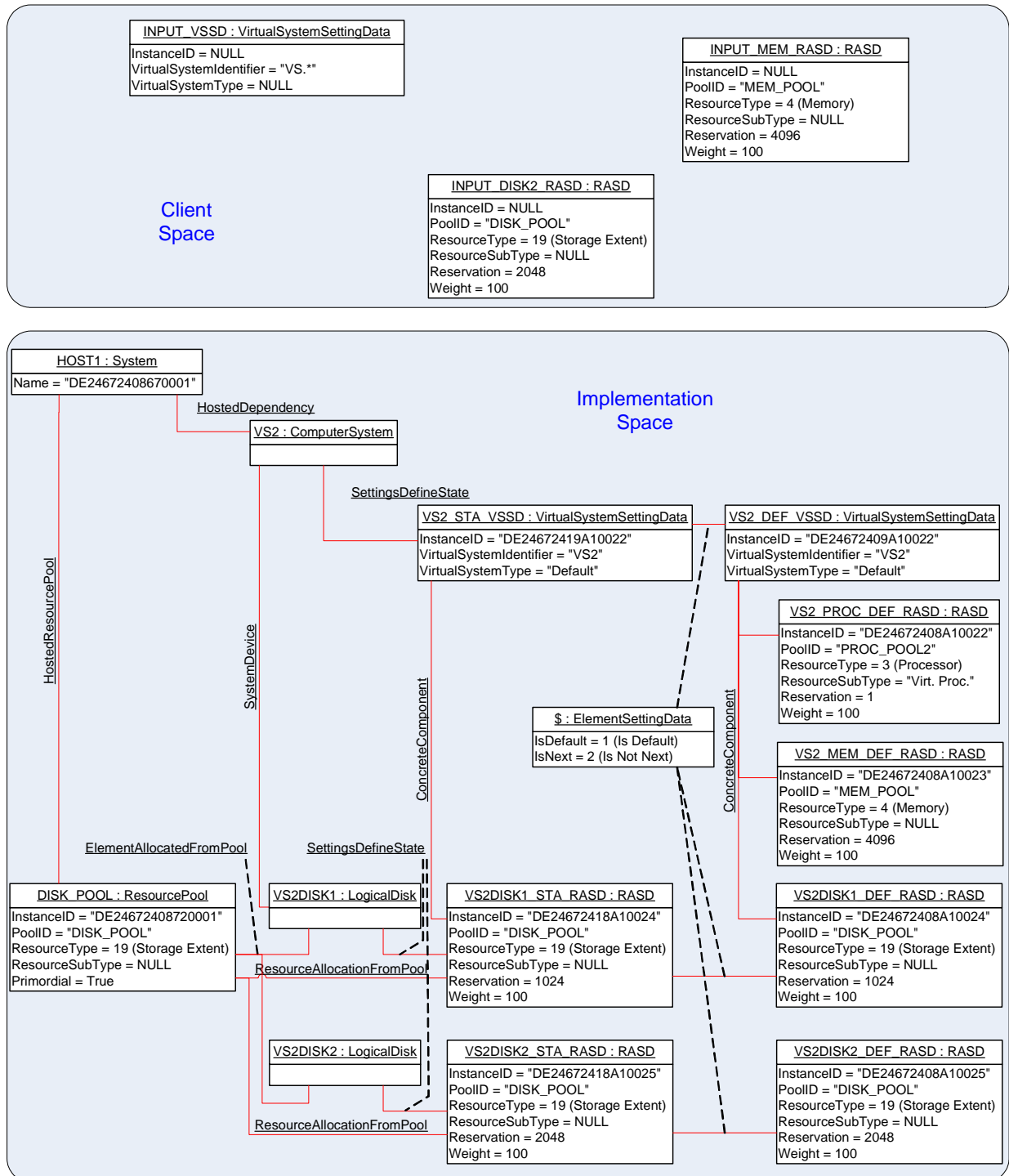
1762 **Assumption:** No assumption is made beyond the general assumption specified in 9.3.

- 1763 1) The client invokes the DefineSystem() method (see 8.2.1) on the virtual system management
 1764 service, as follows.
 - 1765 – The value of the SystemSettings parameter is set to an embedded instance of the
 1766 CIM_VirtualSystemSettingData class.
 - 1767 – The value of the ResourceSettings[] array parameter is set to an array of embedded in-
 1768 stances of the CIM_ResourceAllocationSettingData class.
 - 1769 – The value of the ReferenceConfiguration parameter is set to refer to a “Reference” virtual
 1770 system configuration.
- 1771 2) The implementation executes the DefineSystem() method. The configuration of the new virtual
 1772 system is created according to the client’s requirements. The new virtual system is in the
 1773 “Defined” virtual system state.

1774 The value returned in the ResultingSystem parameter refers to an instance of the
 1775 CIM_ComputerSystem class.

1776 **Result:** The client knows a reference to an instance of the CIM_ComputerSystem class that represents
 1777 the new virtual system.

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



1780
 1781 **Figure 4 – Virtual System Configuration Based on "Input" Virtual System Configurations and**
 1782 **Implementation Defaults**

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

1785 tem. It is based on the “Input” virtual system configuration shown at the top of Figure 4. In this example, it
 1786 is assumed that the ReferenceConfiguration parameter refers to a virtual system configuration that con-
 1787 tains requests for the following resources:

- 1788 • a virtual processor
- 1789 • virtual memory of 1024 MB
- 1790 • a virtual disk of 1024 MB

1791 The “Input” virtual system configuration does not request the allocation of a processor, but because the
 1792 “Reference” virtual configuration does, the resulting virtual system definition contains a request for a
 1793 processor as well.

1794 The input virtual system configuration requests 4096 MB of memory. That value is given preference over
 1795 the value of 1024 that is specified in the “Reference” configuration.

1796 The input virtual system configuration requests a virtual disk in addition to the one requested by the
 1797 “Reference” configuration, resulting in two virtual disks allocated for the new virtual system.

1798 9.3.1.2 Define Virtual System with Implementation-Specific Properties

1799 **Assumption:** No assumption is made beyond the general assumption specified in 9.3.

- 1800 • The client performs use case 9.3.1.1 using an input configuration only. While preparing the input
 1801 virtual system configuration, the client applies use case 9.2.14 to determine the implementation
 1802 class of the CIM_VirtualSystemSettingData class and use case 9.2.15 to determine the various
 1803 implementation classes for the CIM_ResourceAllocationSettingData class for the required
 1804 resource types.

1805 The implementation classes may specify additional properties beyond the set that is defined in
 1806 the respective base classes. The client may use the description information about each of these
 1807 properties that is obtained with the respective class descriptions to request appropriate values
 1808 from end users in order to create valid instances of the implementation class (thereby defining
 1809 implementation-specific resource requirements).

1810 **Result:** The value of the DefinedSystem output parameter refers to an instance of the
 1811 CIM_ComputerSystem class that represents the newly created virtual system. The new system is in the
 1812 “Defined” state.

1813 9.3.2 Virtual System Modification

1814 This clauses describes a set of usecases that modify virtual systems or virtual system configurations.

1815 9.3.2.1 Modify Virtual System State or Definition

1816 **Assumption:** The client knows a reference to an instance of the CIM_ComputerSystem class that
 1817 represents a virtual system.

- 1818 1) The client obtains the instance of the CIM_VirtualSystemSettingData class that represents the
 1819 state or definition of virtual aspects of the affected virtual system (respective use cases are de-
 1820 scribed in the *Virtual System Profile*).
- 1821 2) The client makes conformant changes to the instance of the CIM_VirtualSystemSettingData
 1822 class. In particular, the client must not modify key properties.
- 1823 3) The client invokes the ModifySystemSettings() method (see 8.2.5) on the virtual system
 1824 management service. The value of the SystemSettings parameter is the modified instance from
 1825 step 2).
- 1826 4) The implementation executes the ModifySystemSettings() method, and the configuration of the
 1827 virtual system is modified according to the clients requirements.

1828 **Result:** The requested modification is applied to the state or definition of the virtual system.

1829 9.3.2.2 Add Virtual Resources

1830 **Assumption:** The client knows a reference to an instance of the CIM_VirtualSystemSettingData class
1831 that represents a virtual system configuration.

- 1832 1) The client locally prepares one or more instances of the CIM_ResourceAllocationSettingData
1833 class to represent the resource allocation requests for the new virtual resources.
- 1834 2) The client invokes the AddResourceSettings() method (see 8.2.3) on the virtual system
1835 management service, as follows:
 - 1836 – The value of the AffectedConfiguration parameter is set to refer to the instance of the
1837 CIM_VirtualSystemSettingData class that represents the virtual system configuration that
1838 receives new resources allocations.
 - 1839 – The value of the ResourceSettings[] array parameter is set with each element as one
1840 embedded instance of the CIM_ResourceAllocationSettingData class prepared in step 1).
- 1841 3) The implementation executes the AddResourceSettings() method, adding the requested re-
1842 source allocations and resource allocation requests to the virtual system configuration.

1843 **Result:** The requested resource allocations or resource allocation requests are configured into the refer-
1844 enced virtual system configuration.

1845 9.3.2.3 Modify Virtual Resource State Extension or Virtual Resource Definition

1846 **Assumption:** The client knows references to one or more instances of the CIM_LogicalDevice class that
1847 represent one or more virtual resources.

1848 Alternatively the client knows the reference to an instance of the CIM_ResourceAllocationSettingData
1849 class that represents the virtual resource state extensions or virtual resource definitions. In this case, the
1850 client would obtain the referenced instance by using the intrinsic GetInstance() CIM operation and pro-
1851 ceed with step 4).

- 1852 1) The client invokes the intrinsic Associators() CIM operation for the virtual resource state exten-
1853 sion as follows:
 - 1854 – The value of the ObjectName parameter is set to refer to the instance of the
1855 CIM_LogicalDevice class.
 - 1856 – The value of the AssocClass parameter is set to “CIM_SettingsDefineState”.
 - 1857 – The value of the ResultClass parameter is set to “CIM_ResourceAllocationSettingData”.

1858 The result is a list of instances of the CIM_ResourceAllocationSettingData class. The size of the
1859 list is expected to be 1, and that element represents the virtual resource state extension. If the
1860 client intends to modify the virtual resource state extension, the client skips steps 2) and 3), and
1861 proceeds with step 4). If the client intends to modify the virtual resource definition, the client
1862 continues with step 2).

- 1863 2) The client invokes the intrinsic References() CIM operation for the association instances that
1864 connect the virtual resource definition, as follows:
 - 1865 – The value of the ObjectName parameter is set to refer to the instance of the
1866 CIM_ResourceAllocationSettingData class that was obtained in step 1).
 - 1867 – The value of the ResultClass parameter is set to “CIM_ElementSettingData”.

1868 The result is a list of instances of the CIM_ElementSettingData association that connect various
1869 settings to the virtual resource state extension.

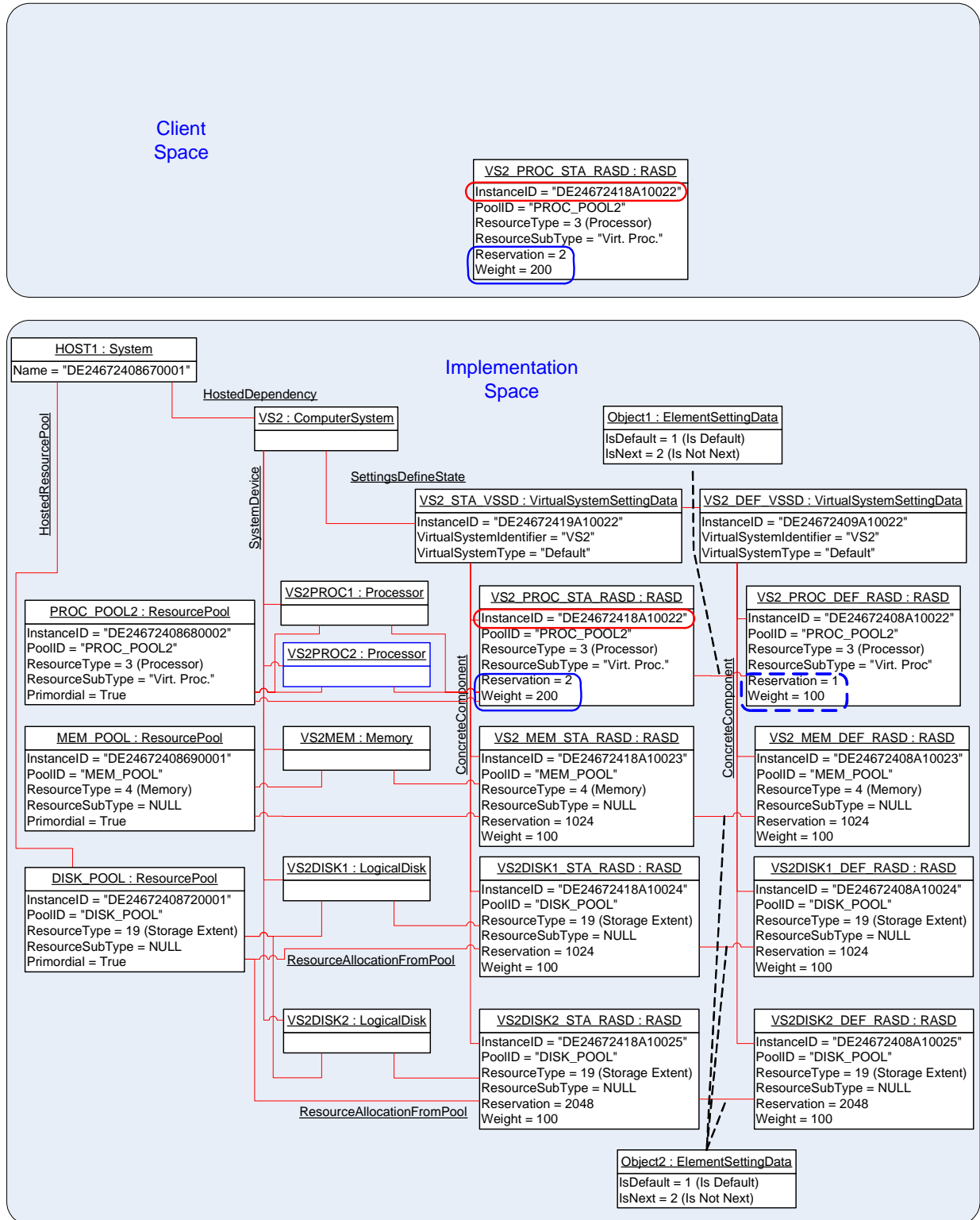
- 1870 3) The client selects from the result set of step 2) the instance in which the IsDefault property has
1871 a value of 1 (Is Default). In that instance, the value of the SettingData property refers to the in-

- 1872 stance of the CIM_ResourceAllocationSettingData class that represents the virtual resource
1873 definition.
- 1874 4) The client invokes the intrinsic GetInstance() CIM operation for the setting that represents the
1875 resource allocation definition. The value of the InstanceName parameter is set to the value of
1876 the SettingData property from the instance of the CIM_ElementSettingData association selected
1877 in step 3).
- 1878 The result is the instance of the CIM_ResourceAllocationSettingData class that represents the
1879 virtual resource definition.
- 1880 5) The client makes conformant changes to the instance of the
1881 CIM_ResourceAllocationSettingData class. In particular, the client must not modify key proper-
1882 ties.
- 1883 Eventually the client executes steps 1) to 5) repetitively, preparing a set of resource allocation
1884 change requests that subsequently are applied as one atomic operation.
- 1885 6) The client invokes the ModifyResourceSettings() method (see 8.2.4) on the virtual system man-
1886 agement service. The values of elements of the ResourceSettings parameter are the modified
1887 instances of the CIM_ResourceAllocationSettingData class that were prepared through repeti-
1888 tive execution of steps in steps 1) to 5).
- 1889 7) The implementation executes the ModifyResourceSettings() method, causing the requested re-
1890 source allocation changes being applied to resource allocation state extensions or resource
1891 allocation definitions.

1892 **Result:** The requested resource modifications are applied to virtual resource state extensions or virtual
1893 resource definitions.

1894 Figure 5 shows the representation of a virtual system. Initially the virtual system was instantiated accord-
1895 ing to the “Defined” virtual system configuration that is show on the right side. During the activation of the
1896 virtual system, required resources were allocated. Virtual resources are represented by instances of sub-
1897 classes of the CIM_LogicalDevice class (CIM_Processor, CIM_Memory, or CIM_LogicalDisk in this case),
1898 with their “State” extensions in the “State” virtual system configuration. Related elements in the virtual sys-
1899 tem representation and the “State” virtual system configuration are associated through instances of the
1900 CIM_SettingsDefineState association.

1901 Entities that are shown in blue color in Figure 5 are involved in the example of a processor resource
1902 modification that is described following the figure.



1903

1904

Figure 5 – Virtual System Resource Modification

1905 Next, the client applied a resource modification on the allocated processor resource within the virtual sys-
 1906 tem's "State" configuration. The "State" configuration is shown to the left of the "Defined" virtual system
 1907 configuration. The client obtained a local copy of the instance of the CIM_ResourceAllocationSettingData
 1908 class that is tagged VS2_PROC_STA_RASD. In that local copy, the client modified the value of the
 1909 Reservation property to 2 and the value of the Weight property to 200. Then the client called the
 1910 ModifyResourceSettings() method with the modified instance as the only element value for the
 1911 ResourceSettings[] array parameter. The execution of that method resulted in another virtual processor
 1912 being allocated to the virtual system.

1913 NOTE: Because a change applied to the "State" virtual system configuration is temporary in nature, a recycling of
 1914 the virtual system will nullify the change and result in a new "State" virtual system configuration based on the
 1915 "Defined" virtual system configuration.

1916 9.3.2.4 Delete Virtual Resources or Virtual Resource Definitions

1917 **Assumption:** The client has references to one or more instances of the
 1918 CIM_ResourceAllocationSettingData class that refer to elements of the "State" or "Defined" virtual system
 1919 configuration of one virtual system. See Clause 9 of the *Virtual System Profile* for respective use cases.

- 1920 1) The client invokes the RemoveResourceSettings() method (see 8.2.6) on the virtual system
 1921 management service. The value of the ResourceSettings[] array parameter is set with each
 1922 element referring to one instance of the CIM_ResourceAllocationSettingData class.
- 1923 2) The implementation executes the RemoveResourceSettings() method. Either all requested re-
 1924 source allocations or resource allocation requests are removed, or none at all.

1925 **Result:** The referenced virtual resources are removed from their respective virtual system configurations.

1926 9.3.3 Destroy Virtual System

1927 **Assumption:** The client knows a reference to an instance of the CIM_ComputerSystem class that repre-
 1928 sents a virtual system (see 9.2.16).

- 1929 1) The client invokes the DestroySystem() method on the virtual system management service.
 1930 The value of the AffectedSystem parameter is set to refer to the instance of the
 1931 CIM_ComputerSystem class that represents the virtual system.
- 1932 2) The implementation executes the DestroySystem() method.

1933 **Result:** The affected virtual system and its virtual resources (together with their definition) are removed
 1934 from the implementation. If the virtual system was in the "Active" state, the "Paused" state, or in the
 1935 "Suspended" state, the running instance of the virtual system and its virtual resources are removed before
 1936 the definition of the virtual system is removed.

1937 NOTE: Dependencies may exist that may prevent the destruction of a virtual system. For example, if definitions or
 1938 instances of other virtual systems refer to elements of the virtual system to be destroyed, the destruction may fail.

1939 9.4 Snapshot-Related Activities

1940 This set of use cases describes activities such as the following:

- 1941 • discovering a virtual system snapshot service
- 1942 • inspecting the capabilities of a virtual system snapshot service
- 1943 • creating a snapshot from a virtual system
- 1944 • applying a snapshot to a virtual system
- 1945 • analyzing a virtual snapshot
- 1946 • analyzing dependencies among snapshots
- 1947 • locating the most recently captured snapshot

- 1948 • destroying a snapshot

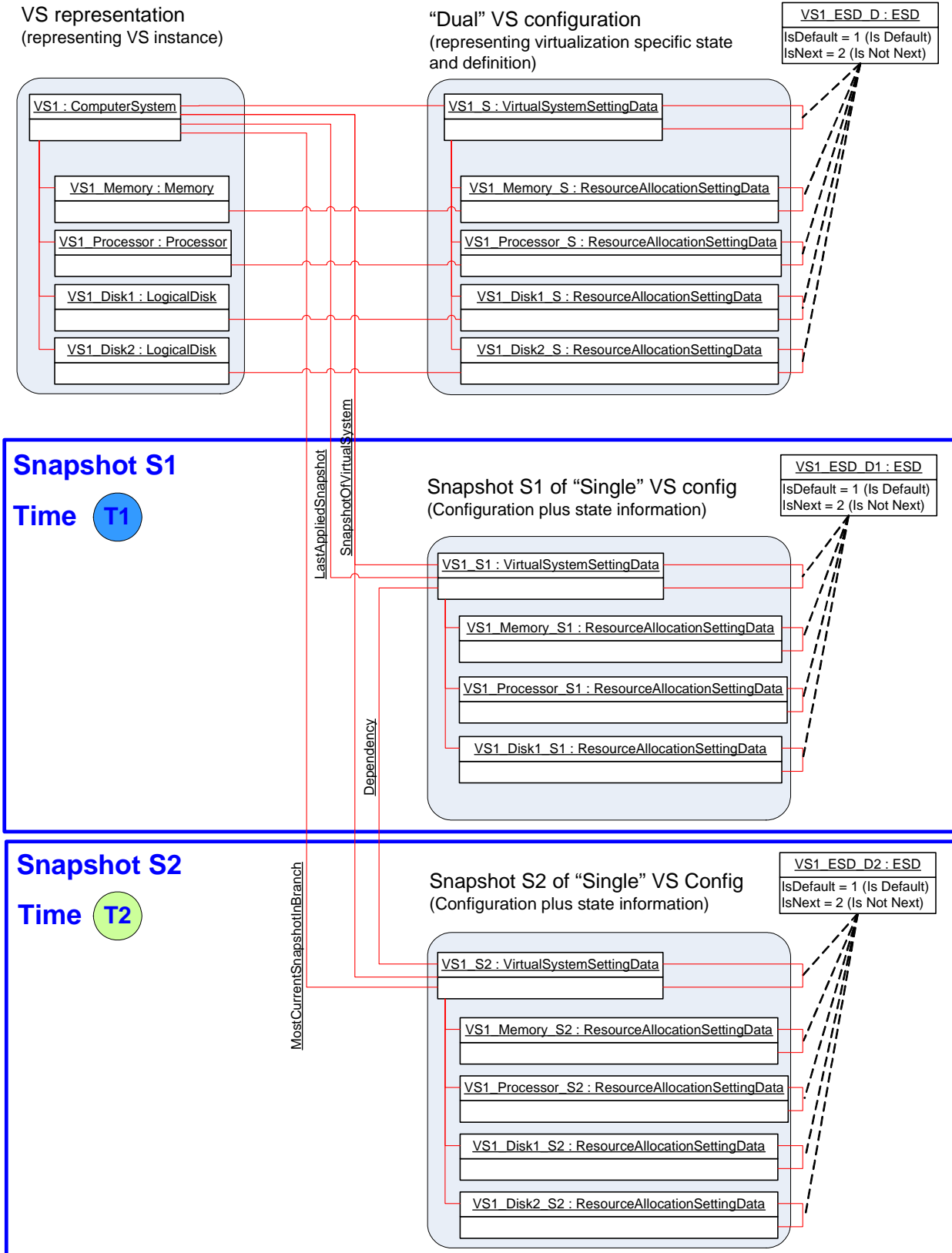
1949 Figure 6 depicts the CIM representation of a virtual system VS1 and of configurations that are associated
1950 with the virtual system at time T3. In the example, it is assumed that the implementation applies the
1951 “Single-Configuration Implementation Approach” as described in the *Virtual System Profile*.

1952 The sequence of events that yield the situation shown in Figure 6 is as follows:

- 1953 1) At time T0, the virtual system VS1 is defined. The initial virtual system definition contains virtual
1954 resource allocation requests for one memory extent, one virtual processor, and one virtual disk.
- 1955 2) At a time after T0 but before T1, the virtual system is activated.
- 1956 3) At time T1, a full snapshot S1 is captured of the virtual system. Virtual system definition and
1957 state are copied into the snapshot. A full snapshot includes the “content” of virtual memory *and*
1958 of virtual disks; a disk snapshot would contain the “content” of virtual disks only.
- 1959 4) The virtual system remains active after the snapshot is captured. The virtual system configura-
1960 tion and the “content” of memory and of virtual disks may change in that interval.
- 1961 5) At a time after T1 but before T2, snapshot S1 is applied to the virtual system, causing definition
1962 and state to be restored to the situation at time T1.
- 1963 6) Still at a time before T2, a second virtual disk is dynamically added to the virtual system. Be-
1964 cause in this example the implementation applies the “Single-Configuration Implementation
1965 Approach,” this change in effect applies to both virtual system definition and virtual system in-
1966 stance and is visible through the “Single” VS configuration.
- 1967 7) At time T2, snapshot S2 is captured of the virtual system. Because at time T2 the virtual system
1968 snapshot S1 is the last applied snapshot, snapshot S2 depends on snapshot S1.
- 1969 8) The virtual system remains active after the snapshot is captured. The virtual system configura-
1970 tion and the “content” of memory and of virtual disks may change in that interval.
- 1971 9) At a time after T2 but before T3, snapshot S2 is applied to the virtual system, causing definition
1972 and state to be restored to the situation at time T2, thereby nullifying changes that were applied
1973 to the virtual system after T2.
- 1974 10) At time T3, the situation is as shown in Figure 6.

1975 **General Assumption:** The client knows the reference to an instance of the
1976 CIM_VirtualSystemSnapshotService class that represents the virtual system snapshot of a host system
1977 (see 9.2.6).

Current Time: T3 > T2 > T1 > T0



1979 **Figure 6 – System Virtualization Profile: Snapshot Example**

1980 **9.4.1 Locate Virtual System Snapshot Service**

1981 **Assumption:** The client knows a reference to an instance of the CIM_System class that represents a
1982 host system that is a central instance of this profile; (see 9.2.4).

- 1983
- The client invokes the intrinsic AssociatorNames() CIM operation as follows:
 - 1984 – The value of the ObjectName parameter is set to refer to the instance of the CIM_System
 - 1985 class.
 - 1986 – The value of the AssocClass parameter is set to “CIM_HostedService”.
 - 1987 – The value of the ResultClass parameter is set to “CIM_VirtualSystemSnapshotService”.

1988 **Result:** The client knows a reference to the instance of the CIM_VirtualSystemSnapshotService class
1989 that represents the virtual system snapshot service serving the host system. If the operation is successful,
1990 the size of the result set is 1.

1991 In the example shown in Figure 3, one instance of the CIM_VirtualSystemSnapshotService class serves
1992 the host system; it is tagged VSSS_1.

1993 **9.4.2 Determine Capabilities of a Virtual System Snapshot Service**

1994 **Assumption:** The client knows a reference to an instance of the CIM_VirtualSystemSnapshotService
1995 class that represents the virtual system snapshot service serving a host system (see 9.4.1).

- 1996 1) The client invokes the intrinsic Associators() CIM operation as follows:
- 1997 – The value of the ObjectName parameter is set to refer to the instance of the
 - 1998 CIM_VirtualSystemSnapshotService class.
 - 1999 – The value of the AssocClass parameter is set to “CIM_ElementCapabilities”.
 - 2000 – The value of the ResultClass parameter is set to
 - 2001 “CIM_VirtualSystemSnapshotServiceCapabilities”.
- 2002 The result is a list of instances of the CIM_VirtualSystemSnapshotServiceCapabilities class. If
2003 the operation is successful, the size of the result set is 1.
- 2004 2) The client analyzes the instance of the CIM_VirtualSystemSnapshotServiceCapabilities class.
- 2005 – The SynchronousMethodsSupported[] array property lists identifiers of methods of the
 - 2006 CIM_VirtualSystemSnapshotServiceCapabilities class that are supported with synchronous
 - 2007 method execution only.
 - 2008 – The AsynchronousMethodsSupported[] array property lists identifiers of methods of the
 - 2009 CIM_VirtualSystemSnapshotServiceCapabilities class that are supported with synchronous
 - 2010 and asynchronous method execution.
 - 2011 – The SnapshotTypesSupported[] array property lists identifiers designating snapshot types
 - 2012 that are supported by the implementation.

2013 **Result:** The client knows the virtual-system-snapshot-related capabilities of the host system in terms of
2014 properties of the CIM_VirtualSystemSnapshotServiceCapabilities class.

2015 In the example shown in Figure 3, one instance of the CIM_VirtualSystemSnapshotServiceCapabilities
2016 class is associated with the host system; it is tagged VSSSC_1.

2017 9.4.3 Create Snapshot

2018 **Assumption:** The client knows a reference to an instance of the CIM_ComputerSystem class that repre-
2019 sents a virtual system hosted by a host system (see 9.2.16). The virtual system is active.

2020 1) The client invokes the CreateSnapshot() method on the virtual system snapshot service, as fol-
2021 lows:

2022 – The value of the AffectedSystem parameter is set to refer to the instance of the
2023 CIM_ComputerSystem class that represents the virtual system.

2024 – The value of the SnapshotType parameter is set to 2 (Full Snapshot).

2025 2) The implementation executes the CreateSnapshot() method.

2026 The value returned in the ResultingSnapshot parameter refers to an instance of the
2027 CIM_VirtualSystemSettingData class that represents the new snapshot.

2028 **Result:** The client knows a reference to the instance of the CIM_VirtualSystemSettingData class that
2029 represents the created virtual system snapshot.

2030 In the example shown in Figure 6, two instances of the CIM_VirtualSystemSettingData class represent
2031 virtual system snapshots S1 and S2 taken at times T1 and T2. Although the situation captured in Figure 6
2032 shows the situation at T3, a snapshot taken at T3 would look identical to S2 (because the current system
2033 at time T3 is unchanged with respect to S2).

2034 9.4.4 Locate Snapshots of a Virtual System

2035 **Assumption:** The client knows a reference to an instance of the CIM_ComputerSystem class that
2036 represents a virtual system (see 9.2.16).

2037 • The client invokes the intrinsic Associators() CIM operation for the list of snapshots, as follows:

2038 – The value of the ObjectName parameter is set to refer to the instance of the
2039 CIM_ComputerSystem class.

2040 – The value of the AssocClass parameter is set to “CIM_SnapshotOfVirtualSystem”.

2041 – The value of the ResultClass parameter is set to “CIM_VirtualSystemSettingData”.

2042 The result is a list of instances of the CIM_VirtualSystemSettingData class.

2043 **Result:** The client knows a set of instances of the CIM_VirtualSystemSettingData class, each represent-
2044 ing a virtual system snapshot taken from the virtual system.

2045 In the example shown in Figure 6, the instances tagged VS_S1 and VS1_S2 of the
2046 CIM_VirtualSystemSettingData class represent snapshots S1 and S2.

2047 9.4.5 Locate the Source Virtual System of a Snapshot

2048 **Assumption:** The client knows the reference to an instance of the CIM_VirtualSystemSettingData class
2049 that represents a virtual system snapshot.

2050 • The client invokes the intrinsic AssociatorNames() CIM operation for the source virtual system
2051 as follows:

2052 – The value of the ObjectName parameter is set to refer to the instance of the
2053 CIM_VirtualSystemSettingData class.

2054 – The value of the AssocClass parameter is set to “CIM_ElementSettingData”.

2055 – The value of the ResultClass parameter is set to “CIM_ComputerSystem”.

2056 The result is a list of references to instances of the CIM_ComputerSystem class. The size of the
2057 list is 1.

2058 **Result:** The client knows a reference to an instance of the CIM_ComputerSystem class that represents
2059 the virtual system that was the source for the snapshot.

2060 NOTE: At this time the present configuration of the virtual system may be completely different from the configuration
2061 that was captured in the snapshot.

2062 In the example shown in Figure 6, the instance of class CIM_ComputerSystem tagged VS1 is the source
2063 of snapshots S1 and S2, represented by instances of the CIM_VirtualSystemSettingData class tagged
2064 VS_S1 and VS_S2.

2065 9.4.6 Locate the Most Current Snapshot in a Branch of Snapshots

2066 **Assumption:** The client knows an instance of the CIM_ComputerSystem class that represents a virtual
2067 system (see 9.2.16).

- 2068 • The client invokes the intrinsic Associators() CIM operation for the most current snapshot in the
2069 current branch of virtual snapshots, as follows:
 - 2070 – The value of the ObjectName parameter is set to refer to the instance of the
2071 CIM_ComputerSystem class.
 - 2072 – The value of the AssocClass parameter is set to “CIM_MostCurrentSnapshotInBranch”.
 - 2073 – The value of the ResultClass parameter is set to “CIM_VirtualSystemSettingData”.
- 2074 The result is a list of instances of the CIM_VirtualSystemSettingData class. The size of the list is
2075 1.

2076 **Result:** The client knows an instance of the CIM_VirtualSystemSettingData class that represents the vir-
2077 tual system snapshot that is the most current snapshot in the current branch of snapshots.

2078 In the example shown in Figure 6, the instance of the CIM_VirtualSystemSettingData class that is tagged
2079 VS1_2 represents the most current snapshot in the current branch of snapshots. This is the case because
2080 that snapshot was applied most recently to the virtual system and no other snapshot was applied to or
2081 created from the virtual system since then.

2082 9.4.7 Locate Dependent Snapshots

2083 **Assumption:** The client knows a reference to an instance of the CIM_VirtualSystemSettingData class
2084 that represents a virtual system snapshot (see 9.4.4).

- 2085 • The client invokes the intrinsic AssociatorNames() CIM operation for the list of dependent snap-
2086 shots as follows:
 - 2087 – The value of the ObjectName parameter is set to refer to the instance of the
2088 CIM_VirtualSystemSettingData class.
 - 2089 – The value of the AssocClass parameter is set to “CIM_Dependency”.
 - 2090 – The value of the ResultClass parameter is set to “CIM_VirtualSystemSettingData”.
 - 2091 – The value of the Role parameter is set to “Antecedent”.
 - 2092 – The value of the ResultRole parameter is set to “Dependent”.
- 2093 The result is a list of references to instances of the CIM_VirtualSystemSettingData class.

2094 **Result:** The client knows a set of instances of the CIM_VirtualSystemSettingData class that represent vir-
2095 tual system snapshots that depend on the input virtual system snapshot. The set may be empty, indicating
2096 that no dependent snapshots exist.

2097 In the example shown in Figure 6, the instance tagged VS_S2 represents snapshot S2, which is depend-
2098 ent on snapshot S1, which is represented by the instance tagged VS_S1.

2099 9.4.8 Locate Parent Snapshot

2100 **Assumption:** The client knows a reference to an instance of the CIM_VirtualSystemSettingData class
2101 that represents a virtual system snapshot (see 9.4.4).

2102 • The client invokes the intrinsic AssociatorNames() CIM operation for the parent snapshot as fol-
2103 lows:

2104 – The value of the ObjectName parameter is set to refer to the instance of the
2105 CIM_VirtualSystemSettingData class that represents the virtual system snapshot.

2106 – The value of the AssocClass parameter is set to “CIM_Dependency”.

2107 – The value of the ResultClass parameter is set to “CIM_VirtualSystemSettingData”.

2108 – The value of the Role parameter is set to “Dependent”.

2109 – The value of the ResultRole parameter is set to “Antecedent”.

2110 The result is a list of references to instances of the CIM_VirtualSystemSettingData class that
2111 represent virtual system snapshots. The list has a size of 1 or 0.

2112 **Result:** The client knows the instance of the CIM_VirtualSystemSettingData class that represents the par-
2113 ent virtual system snapshot of the input virtual system snapshot. The set may be empty, indicating that no
2114 parent snapshots exist.

2115 In the example shown in Figure 6, the instance tagged VS_S1 represents snapshot S1, which is the par-
2116 ent of snapshot S2, which is represented by the instance tagged VS_S2.

2117 9.4.9 Apply Snapshot

2118 **Assumption:** The client knows a reference to an instance of the CIM_VirtualSystemSettingData class
2119 that represents a virtual system snapshot (see 9.4.3 or 9.4.4). The client knows a reference to the in-
2120 stance of the CIM_ComputerSystem class that represents the virtual system that was the source for the
2121 snapshot (see 9.4.5). The virtual system is active.

2122 1) The client invokes the ApplySnapshot() method on the virtual system snapshot service. The
2123 value of the Snapshot parameter is set to refer to the instance of the
2124 CIM_VirtualSystemSettingData class that represents the snapshot.

2125 2) The snapshot is applied into the active virtual system as follows:

2126 a) The virtual system is deactivated. This implies a disruptive termination of the software that
2127 may be active in the instance of the virtual system.

2128 b) The virtual system is reconfigured according to the virtual system snapshot. For a disk
2129 snapshot, this applies to the disk resources only.

2130 c) If the applied snapshot is a full snapshot, all stateful resources like memory and disk are
2131 restored to the situation that was captured in the snapshot. If the applied snapshot is a disk
2132 snapshot, only disk resources are restored.

2133 d) The virtual system is activated. If the applied snapshot is a full snapshot, the virtual system
2134 starts from the situation that was captured by the full snapshot. If the applied snapshot was
2135 a disk snapshot, a normal virtual system activation occurs.

2136 **Result:** The virtual system is restored to the situation that was in place when the snapshot was taken.

2137 In the example shown in Figure 6, the situation is depicted at time T3, immediately after the activation of
2138 snapshot S2 within virtual system VS1.

2139 **9.4.10 Destroy Snapshot**

2140 **Assumption:** The client knows the reference to an instance of the CIM_VirtualSystemSettingData class
2141 that represents a virtual system snapshot (see 9.2.16).

2142 1) The client invokes the DestroySnapshot() method on the virtual system management service.
2143 The value of the Snapshot parameter is set to refer to the instance of the
2144 CIM_VirtualSystemSettingData class that represents the snapshot.

2145 2) The snapshot is removed from the implementation.

2146 **Result:** The snapshot no longer exists within the implementation.

2147 **10 CIM Elements**

2148 Table 31 lists CIM elements that are defined or specialized for this profile. Each CIM element shall be
2149 implemented as described in Table 31. The CIM Schema descriptions for any referenced element and its
2150 sub-elements apply.

2151 Clauses 7 ("Implementation") and 8 ("Methods") may impose additional requirements on these elements.

2152 **Table 31 – CIM Elements: System Virtualization Profile**

Element Name	Requirement	Description
CIM_AffectedJobElement	Conditional	See 10.1.
CIM_ConcreteJob	Conditional	See 10.2.
CIM_Dependency	Conditional	See 10.3.
CIM_ElementCapabilities (Host System)	Mandatory	See 10.4.
CIM_ElementCapabilities (Virtual System Management Service)	Mandatory	See 10.5.
CIM_ElementCapabilities (Virtual System Snapshot Service)	Conditional	See 10.6.
CIM_ElementCapabilities (Snapshots of Virtual Systems)	Conditional	See 10.7.
CIM_ElementConformsToProfile	Mandatory	See 10.8.
CIM_HostedDependency	Mandatory	See 10.9.
CIM_HostedService (Virtual System Management Service)	Conditional	See 10.10.
CIM_HostedService (Virtual System Snapshot Service)	Conditional	See 10.11.
CIM_LastAppliedSnapshot	Conditional	See 10.12.
CIM_MostCurrentSnapshotInBranch	Conditional	See 10.13.
CIM_ReferencedProfile	Conditional	See 10.14.
CIM_RegisteredProfile	Mandatory	See 10.15.
CIM_ServiceAffectsElement (Virtual System Management Service)	Conditional	See 10.16.
CIM_ServiceAffectsElement (Virtual System Snapshot Service)	Conditional	See 10.17.
CIM_SnapshotOfVirtualSystem	Conditional	See 10.18.
CIM_System	Mandatory	See 10.19.
CIM_VirtualSystemManagementCapabilities	Mandatory	See 10.20.
CIM_VirtualSystemManagementService	Conditional	See 10.21.

Element Name	Requirement	Description
CIM_VirtualSystemSettingData (Input)	Conditional	See 10.22.
CIM_VirtualSystemSettingData (Snapshot)	Conditional	See 10.23.
CIM_VirtualSystemSnapshotCapabilities	Conditional	See 10.24.
CIM_VirtualSystemSnapshotService	Optional	See 10.25.
CIM_VirtualSystemSnapshotServiceCapabilities	Conditional	See 10.26.

2153 **10.1 CIM_AffectedJobElement (Conditional)**

2154 The implementation of the CIM_AffectedJobElement association is conditional with respect to the support
 2155 of a non-NULL value for at least one element of the AsynchronousMethodsSupported[] array property of
 2156 the CIM_VirtualSystemManagementCapabilities class.

2157 An implementation shall use the CIM_AffectedJobElement association to associate an instance of the
 2158 CIM_ConcreteJob class that represents an asynchronous task and an instance of the
 2159 CIM_ComputerSystem class that represents a virtual system that is affected by its execution.

2160 Table 32 contains the requirements for elements of this association.

2161 **Table 32 – Association: CIM_AffectedJobElement**

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

2162 **10.2 CIM_ConcreteJob (Conditional)**

2163 The implementation of the CIM_ConcreteJob class is conditional with respect to the support of a non-
 2164 NULL value for at least one element of the AsynchronousMethodsSupported[] array property of the
 2165 CIM_VirtualSystemManagementCapabilities class.

2166 An implementation shall use an instance of the CIM_ConcreteJob class to represent an asynchronous
 2167 task.

2168 Table 33 contains requirements for elements of this class.

2169 **Table 33 – Class: CIM_ConcreteJob**

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

2170 10.3 CIM_Dependency (Conditional)

2171 The implementation of the CIM_Dependency association is conditional with respect to the support of vir-
2172 tual system snapshots (see 7.7.1.1).

2173 An implementation shall use an instance of the CIM_Dependency association to associate an instance of
2174 the CIM_VirtualSystemSettingData class that represents a parent snapshot and an instance of the
2175 CIM_VirtualSystemSettingData class that represents a dependent snapshot.

2176 Table 34 contains requirements for elements of this class.

2177 **Table 34 – Class: CIM_Dependency Class**

Elements	Requirement	Notes
Antecedent	Mandatory	Key: Reference to an instance of the CIM_VirtualSystemSettingData class that represents a parent snapshot Cardinality: 0..1
Dependent	Mandatory	Key: Reference to an instance of the CIM_VirtualSystemSettingData class that represents a dependent snapshot Cardinality: 0..1

2178 10.4 CIM_ElementCapabilities (Host System)

2179 An implementation shall use an instance of the CIM_ElementCapabilities association to associate an in-
2180 stance of the CIM_System class that represents a host system with an instance of the
2181 CIM_VirtualSystemManagementCapabilities class that describes the virtual system management capabili-
2182 ties of the host system.

2183 Table 35 contains requirements for elements of this association.

2184 **Table 35 – Association: CIM_ElementCapabilities (Host System)**

Elements	Requirement	Notes
ManagedElement	Mandatory	Key: Reference to instance of the CIM_System class that represents a host system Cardinality: 1
Capabilities	Mandatory	Key: Reference to an instance of the CIM_VirtualSystemManagementCapabilities class that describes the capabilities of a host system Cardinality: 1

2185 10.5 CIM_ElementCapabilities (Virtual System Management Service) 2186 (Conditional)

2187 The implementation of the CIM_ElementCapabilities association for the virtual system management
2188 service is conditional with respect to the support of any of:

- 2189 • virtual system definition and destruction (see 7.4.6.1)
- 2190 • virtual resource addition and removal (see 7.4.6.2)
- 2191 • virtual system and resource modification (see 7.4.6.3)

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

2196 Table 36 contains requirements for elements of this association.

2197 **Table 36 – Association: CIM_ElementCapabilities (Virtual System Management)**

Elements	Requirement	Notes
ManagedElement	Mandatory	Key: Reference to instance of the CIM_VirtualSystemManagementService class Cardinality: 0..1
Capabilities	Mandatory	Key: Reference to an instance of the CIM_VirtualSystemManagementCapabilities class Cardinality: 1

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

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

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

2205 Table 37 contains requirements for elements of this association.

2206 **Table 37 – Association: CIM_ElementCapabilities (Snapshot Service)**

Elements	Requirement	Notes
ManagedElement	Mandatory	Key: Reference to an instance of the CIM_VirtualSystemSnapshotService class that repre- sents a virtual system snapshot service Cardinality: 1
Capabilities	Mandatory	Key: Reference to the instance of the CIM_VirtualSystemSnapshotServiceCapabilities class that represents the capabilities of the virtual system snapshot service Cardinality: 1

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

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

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

2213 Table 38 contains requirements for elements of this association.

2214

Table 38 – Association: CIM_ElementCapabilities (Snapshots of Virtual Systems)

Elements	Requirement	Notes
ManagedElement	Mandatory	Key: Reference to an instance of the CIM_ComputerSystem class that represents a virtual system Cardinality: *
Capabilities	Mandatory	Key: Reference to the instance of the CIM_VirtualSystemSnapshotCapabilities class that describes the current applicability of snapshot related services to the virtual system Cardinality: 1

2215 10.8 CIM_ElementConformsToProfile

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

2220 Table 39 contains requirements for elements of this association.

2221

Table 39 – Association: CIM_ElementConformsToProfile

Elements	Requirement	Notes
ConformantStandard	Mandatory	Key: Reference to an instance of the CIM_Registered-Profile class that represents an implementation of this profile Cardinality: 1
ManagedElement	Mandatory	Key: Reference to an instance of the CIM_System class that represents a host system Cardinality: *

2222 10.9 CIM_HostedDependency

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

2226 Table 40 contains requirements for elements of this association.

2227

Table 40 – Association: CIM_HostedDependency

Elements	Requirement	Notes
Antecedent	Mandatory	Key: Reference to an instance of the CIM_System class that represents a host system Cardinality: 1
Dependent	Mandatory	Key: Reference to an instance of the CIM_ComputerSystem class that represents a virtual system Cardinality: *

2228 **10.10 CIM_HostedService (Virtual System Management Service) (Conditional)**

2229 The implementation of the CIM_HostedService association for the virtual system management service is
 2230 conditional with respect to the support of any of:

- 2231 • virtual system definition and destruction (see 7.4.6.1)
- 2232 • virtual resource addition and removal (see 7.4.6.2)
- 2233 • virtual system and resource modification (see 7.4.6.3)

2234 The implementation shall use an instance of the CIM_HostedService association to associate an instance
 2235 of the CIM_System class that represents a host system and the instance of the CIM_VirtualSystem-
 2236 ManagementService class that represents the virtual system management service that is hosted by a
 2237 host system.

2238 Table 41 contains requirements for elements of this association.

2239 **Table 41 – Association: CIM_HostedService (Virtual System Management Service)**

Elements	Requirement	Notes
Antecedent	Mandatory	Key: Reference to an instance of the CIM_System class that represents a host system Cardinality: 1
Dependent	Mandatory	Key: Reference to an instance of the CIM_VirtualSystemManagementService class that represents a virtual system management service Cardinality: 0..1

2240 **10.11 CIM_HostedService (Virtual System Snapshot Service) (Conditional)**

2241 The implementation of the CIM_HostedService association is conditional with respect to the support of
 2242 virtual system snapshots (see 7.7.1.1).

2243 The implementation shall use an instance of the CIM_HostedService association to associate an instance
 2244 of the CIM_ComputerSystem class that represents a host system and the instance of the
 2245 CIM_VirtualSystemSnapshotService class that represents the virtual system snapshot service.

2246 Table 42 contains requirements for elements of this association.

2247 **Table 42 – Association: CIM_HostedService (Virtual System Snapshot Service)**

Elements	Requirement	Notes
Antecedent	Mandatory	Key: Reference to an instance of the CIM_System class that represents a host system Cardinality: 1
Dependent	Mandatory	Key: Reference to an instance of the CIM_VirtualSystemSnapshotService class that represents a virtual system snapshot service Cardinality: 0..1

2248 10.12 CIM_LastAppliedSnapshot (Conditional)

2249 The implementation of the CIM_LastAppliedSnapshot association is conditional with respect to the sup-
2250 port of virtual system snapshots (see 7.7.1.1).

2251 An implementation shall use an instance of the CIM_LastAppliedSnapshot association to associate an in-
2252 stance of the CIM_ComputerSystem class that represents a virtual system and the instance of the
2253 CIM_VirtualSystemSettingData class that represents the virtual system snapshot that was last applied to
2254 the virtual system.

2255 Table 43 contains requirements for elements of this association.

2256 **Table 43 – Association: CIM_LastAppliedSnapshot**

Elements	Requirement	Notes
Antecedent	Mandatory	Key: Reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot Cardinality: 0..1
Dependent	Mandatory	Key: Reference to the instance of the CIM_ComputerSystem class that represents the virtual system Cardinality: 0..1

2257 10.13 CIM_MostCurrentSnapshotInBranch (Conditional)

2258 The implementation of the CIM_MostCurrentSnapshotInBranch association is conditional with respect to
2259 the support of virtual system snapshots (see 7.7.1.1).

2260 An implementation shall use an instance of the CIM_MostCurrentSnapshotInBranch association to
2261 associate an instance of the CIM_ComputerSystem class that represents a virtual system and the
2262 instance of the CIM_VirtualSystemSettingData class that represents the most current snapshot in a
2263 branch of virtual system snapshots. The most current snapshot in a branch of snapshots related to an in-
2264 stance of a virtual system is the younger of the following snapshots:

- 2265 • the snapshot that was most recently captured from the virtual system instance
- 2266 • the snapshot that was last applied to the instance

2267 Table 44 contains requirements for elements of this association.

2268 **Table 44 – Association: CIM_MostCurrentSnapshotInBranch**

Elements	Requirement	Notes
Antecedent	Mandatory	Key: Reference to the instance of the CIM_ComputerSystem class that represents the virtual system Cardinality: 0..1
Dependent	Mandatory	Key: Reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot Cardinality: 0..1

2269 **10.14 CIM_ReferencedProfile (Conditional)**

2270 The implementation of the CIM_ReferencedProfile association is conditional with respect to the support of
 2271 resource allocation DMTF management profiles like the *Generic Device Resource Virtualization Profile*.

2272 An implementation shall use an instance of the CIM_ReferencedProfile association to associate an in-
 2273 stance of the CIM_RegisteredProfile class that represents an implementation of this profile and any
 2274 instance of the CIM_RegisteredProfile class that represents an implementation of a resource allocation
 2275 DMTF management profile that describes virtual resource allocation that is implemented by the
 2276 implementation.

2277 Table 45 contains requirements for elements of this association.

2278 **Table 45 – Association: CIM_ReferencedProfile**

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

2279 **10.15 CIM_RegisteredProfile**

2280 An implementation shall use an instance of the CIM_RegisteredProfile class to represent an
 2281 implementation of this profile.

2282 Table 46 contains requirements for elements of this class.

2283 **Table 46 – Class: CIM_RegisteredProfile**

Elements	Requirement	Notes
InstanceID	Mandatory	Key
RegisteredOrganization	Mandatory	Shall be set to "DMTF".
RegisteredName	Mandatory	Shall be set to "System Virtualization".
RegisteredVersion	Mandatory	Shall be set to the version of this profile ("1.0.0").

2284 **10.16 CIM_ServiceAffectsElement (Virtual System Management Service)**
 2285 **(Conditional)**

2286 The implementation of the CIM_ServiceAffectsElement association for the virtual system management
 2287 service is conditional with respect to the support of any of:

- 2288 • virtual system definition and destruction (see 7.4.6.1)
- 2289 • virtual resource addition and removal (see 7.4.6.2)
- 2290 • virtual system and resource modification (see 7.4.6.3)

2291 The implementation shall use an instance of the CIM_ServiceAffectsElement association to associate an
 2292 instance of the CIM_VirtualSystemManagementService class that represents a virtual system manage-

2293 ment service and any instance of the CIM_ComputerSystem class that represents a virtual system that is
 2294 managed by that virtual system management service.

2295 Table 47 contains requirements for elements of this association.

2296 **Table 47 – Association: CIM_ServiceAffectsElement (Virtual System Management Service)**

Elements	Requirement	Notes
AffectedElement	Mandatory	Key: Reference to instance of the CIM_ComputerSystem class that represents a managed virtual system Cardinality: *
AffectingElement	Mandatory	Key: Reference to an instance of the CIM_VirtualSystemManagementService class that represents a virtual system management service Cardinality: 0..1

2297 **10.17 CIM_ServiceAffectsElement (Virtual System Snapshot Service)**
 2298 **(Conditional)**

2299 The implementation of the CIM_ServiceAffectsElement association is conditional with respect to the sup-
 2300 port of virtual system snapshots (see 7.7.1.1).

2301 The implementation shall use an instance of the CIM_ServiceAffectsElement association to associate an
 2302 instance of the CIM_VirtualSystemSnapshotService class that represents a virtual system management
 2303 service with the following instances:

- 2304 • any instance of the CIM_ComputerSystem class that represents a virtual system that is man-
 2305 aged by that virtual system management service
- 2306 • any instance of the CIM_VirtualSystemSettingData class that represents a virtual system snap-
 2307 shot

2308 Table 48 contains requirements for elements of this association.

2309 **Table 48 – Association: CIM_ServiceAffectsElement**

Elements	Requirement	Notes
AffectedElement	Mandatory	Key: Reference to instance of the CIM_ComputerSystem class that represents a virtual system or the CIM_VirtualSystemSettingData class that represents a managed snapshot Cardinality: *
AffectingElement	Mandatory	Key: Reference to an instance of the CIM_VirtualSystemManagementService class that represents a virtual system snapshot service Cardinality: 0..1

2310 **10.18 CIM_SnapshotOfVirtualSystem (Conditional)**

2311 The implementation of the CIM_SnapshotOfVirtualSystem association is conditional with respect to the
 2312 support of virtual system snapshots (see 7.7.1.1).

2313 An implementation shall use an instance of the CIM_SnapshotOfVirtualSystem association to associate
 2314 an the instance of the CIM_ComputerSystem class that represents the virtual system that was the source

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

2317 Table 49 contains requirements for elements of this association.

2318 **Table 49 – Association: CIM_SnapshotOfVirtualSystem**

Elements	Requirement	Notes
Antecedent	Mandatory	Key: Reference to the instance of the CIM_ComputerSystem class that represents the source virtual system Cardinality: 0..1
Dependent	Mandatory	Key: Reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot Cardinality: *

2319 **10.19 CIM_System**

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

2321 Table 51 contains requirements for elements of this class.

2322 **Table 50 – Class: CIM_VirtualSystemManagementCapabilities**

Elements	Requirement	Notes
CreationClassName	Mandatory	Key.
Name	Mandatory	Key

2323 **10.20 CIM_VirtualSystemManagementCapabilities**

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

2326 Table 51 contains requirements for elements of this class.

2327 **Table 51 – Class: CIM_VirtualSystemManagementCapabilities**

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

2328 **10.21 CIM_VirtualSystemManagementService (Conditional)**

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

- 2331 • virtual system definition and destruction (see 7.4.6.1)

2348 **10.23 CIM_VirtualSystemSettingData (Snapshot) (Conditional)**

2349 The implementation of the CIM_VirtualSystemSettingData class for the representation of snapshots of vir-
2350 tual systems is conditional with respect to the support of virtual system snapshots (see 7.7.1.1).

2351 An instance of the CIM_VirtualSystemSettingData class shall be used to represent snapshots of virtual
2352 systems.

2353 Table 54 contains requirements for elements of this class.

2354 **Table 54 – Class: CIM_VirtualSystemSettingData (Snapshot)**

Elements	Requirement	Notes
InstanceID	Mandatory	Key
Caption	Optional	See CIM Schema.
Description	Optional	See CIM Schema.
ElementName	Optional	See CIM Schema.
VirtualSystemIdentifier	Optional	See CIM Schema.
VirtualSystemType	Optional	See CIM Schema.
Notes	Optional	See CIM Schema.
CreationTime	Mandatory	The value shall reflect the creation time of the snapshot.
ConfigurationID	Optional	See CIM Schema.
ConfigurationDataRoot	Optional	See CIM Schema.
ConfigurationFile	Mandatory	This element shall have a value of NULL.
SnapshotDataRoot	Mandatory	This element shall have a value of NULL.
SuspendDataRoot	Optional	See CIM Schema.
SwapFileDataRoot	Mandatory	This element shall have a value of NULL.
LogDataRoot	Optional	See CIM Schema.
AutomaticStartupAction	Mandatory	This element shall have a value of NULL.
AutomaticStartupActionDelay	Mandatory	This element shall have a value of NULL.
AutomaticStartupActionSequenceNumber	Mandatory	This element shall have a value of NULL.
AutomaticShutdownAction	Mandatory	This element shall have a value of NULL.
AutomaticRecoveryAction	Mandatory	This element shall have a value of NULL.
RecoveryFile	Mandatory	This element shall have a value of NULL.
NOTE: Elements marked as mandatory but with a required value of NULL shall in effect not be implemented. Respective information applies to the virtual system as a whole, not just to a particular snapshot, and is covered by the instance of the CIM_VirtualSystemSettingData class in the "State" and the "Defined" virtual system configuration.		

2355 **10.24 CIM_VirtualSystemSnapshotCapabilities (Optional)**

2356 The implementation of the optional CIM_VirtualSystemSnapshotCapabilities class is specified only if vir-
2357 tual system snapshots are supported; (see 7.7.1.1).

2358 An instance of the CIM_VirtualSystemSnapshotCapabilities class may be used to represent the current
2359 applicability of snapshot-related services to one virtual system.

2360 Table 55 contains requirements for elements of this class.

2361 **Table 55 – Class: CIM_VirtualSystemSnapshotCapabilities**

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

2362 10.25 CIM_VirtualSystemSnapshotService (Optional)

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

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

2367 Table 56 contains requirements for elements of this class.

2368 **Table 56 – Class: CIM_VirtualSystemSnapshotService**

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

2369 10.26 CIM_VirtualSystemSnapshotServiceCapabilities (Conditional)

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

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

2374 Table 57 contains requirements for elements of this class.

2375 **Table 57 – Class: CIM_VirtualSystemSnapshotServiceCapabilities**

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

2376

2377
2378
2379
2380

ANNEX A (Informative)

Change Log

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

ANNEX B (Informative)

2381
2382
2383
2384

Acknowledgements

2385 The authors wish to acknowledge the following people.

2386 Editor:

- 2387 • Michael Johanssen – IBM

2388 Contributors:

- 2389 • Gareth Bestor – IBM
- 2390 • Chris Brown – HP
- 2391 • Mike Dutch – Symantec
- 2392 • Jim Fehlig – Novell
- 2393 • Kevin Fox – Sun Microsystems, Inc.
- 2394 • Sue Gnat - cPubs
- 2395 • Ron Goering – IBM
- 2396 • Daniel Hiltgen – EMC/VMware
- 2397 • Kelly Holcomb - cPubs
- 2398 • Michael Johanssen – IBM
- 2399 • Larry Lamers – EMC/VMware
- 2400 • Andreas Maier – IBM
- 2401 • Aaron Merkin – IBM
- 2402 • John Parchem – Microsoft
- 2403 • Joanne Saathof - cPubs
- 2404 • Nihar Shah – Microsoft
- 2405 • David Simpson – IBM
- 2406 • Carl Waldspurger – EMC/VMware