



1

2

3

4

**Document Number: DSP1042**

**Date: 2010-04-22**

**Version: 1.0.0**

5 **System Virtualization Profile**

6 **Document Type: Specification**

7 **Document Status: DMTF Standard**

8 **Document Language: E**

## 9 Copyright Notice

10 Copyright © 2007, 2010 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, provided that correct attribution is given. As DMTF specifications may be revised from time  
14 to time, the particular version and release date should always be noted.

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25 withdrawn or modified after publication, and shall be indemnified and held harmless by any party  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

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

# CONTENTS

|    |       |   |    |
|----|-------|---|----|
| 32 | 1     | Scope .....   | 9  |
| 33 | 2     | Normative references .....  | 9  |
| 34 | 3     | Terms and definitions .....   | 10 |
| 35 | 4     | Symbols and abbreviated terms.....                                      | 11 |
| 36 | 5     | Synopsis.....   | 12 |
| 37 | 6     | Description .....   | 13 |
| 38 | 6.1   | Profile relationships.....  | 13 |
| 39 | 6.2   | System virtualization class schema.....                                 | 15 |
| 40 | 6.3   | Virtual system configurations .....                                     | 17 |
| 41 | 6.4   | Resource allocation .....   | 18 |
| 42 | 6.5   | Snapshots .....   | 19 |
| 43 | 7     | Implementation.....   | 19 |
| 44 | 7.1   | Host system .....   | 19 |
| 45 | 7.2   | Profile registration.....   | 19 |
| 46 | 7.2.1 | This profile .....  | 20 |
| 47 | 7.2.2 | Scoped resource allocation profiles.....                                | 20 |
| 48 | 7.3   | Representation of hosted virtual systems.....                           | 20 |
| 49 | 7.3.1 | Profile conformance for hosted virtual systems.....                     | 21 |
| 50 | 7.3.2 | CIM_VirtualSystemSettingData.VirtualSystemType property .....           | 21 |
| 51 | 7.4   | Virtual system management capabilities .....                            | 21 |
| 52 | 7.4.1 | CIM_VirtualSystemManagementCapabilities class .....                     | 21 |
| 53 | 7.4.2 | CIM_VirtualSystemManagementCapabilities.VirtualSystemTypesSupported[ ]  |    |
| 54 |       | array property.....   | 21 |
| 55 | 7.4.3 | CIM_VirtualSystemManagementCapabilities.SynchronousMethodsSupported[ ]  |    |
| 56 |       | array property.....   | 21 |
| 57 | 7.4.4 | CIM_VirtualSystemManagementCapabilities.AsynchronousMethodsSupported[ ] |    |
| 58 |       | array property.....   | 22 |
| 59 | 7.4.5 | CIM_VirtualSystemManagementCapabilities.IndicationsSupported[ ] array   |    |
| 60 |       | property.....   | 22 |
| 61 | 7.4.6 | Grouping Rules for implementations of methods of the                    |    |
| 62 |       | CIM_VirtualSystemManagementService class.....                           | 22 |
| 63 | 7.5   | Virtual system definition and modification.....                         | 23 |
| 64 | 7.5.1 | CIM_VirtualSystemSettingData.InstanceID property.....                   | 23 |
| 65 | 7.5.2 | CIM_VirtualSystemSettingData.ElementName property .....                 | 23 |
| 66 | 7.5.3 | CIM_VirtualSystemSettingData.VirtualSystemIdentifier property .....     | 24 |
| 67 | 7.5.4 | CIM_VirtualSystemSettingData.VirtualSystemType property .....           | 24 |
| 68 | 7.6   | Virtual resource definition and modification .....                      | 24 |
| 69 | 7.7   | Virtual system snapshots.....   | 25 |
| 70 | 7.7.1 | Virtual system snapshot service and capabilities .....                  | 25 |
| 71 | 7.7.2 | Virtual system snapshot representation.....                             | 26 |
| 72 | 7.7.3 | Designation of the last applied snapshot.....                           | 26 |
| 73 | 7.7.4 | Designation of the most current snapshot in branch .....                | 26 |
| 74 | 7.7.5 | Virtual system snapshot capabilities .....                              | 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 CIM_VirtualSystemManagementService class.....            | 29 |
| 81 | 8.2.1 | CIM_VirtualSystemManagementService.DefineSystem( ) method.....          | 29 |
| 82 | 8.2.2 | CIM_VirtualSystemManagementService.DestroySystem( ) method.....         | 31 |

|     |        |   |    |
|-----|--------|---|----|
| 83  | 8.2.3  | CIM_VirtualSystemManagementService.AddResourceSettings( ) method                  |    |
| 84  |        | (Conditional).....  | 32 |
| 85  | 8.2.4  | CIM_VirtualSystemManagementService.ModifyResourceSettings( ) method .....         | 34 |
| 86  | 8.2.5  | CIM_VirtualSystemManagementService.ModifySystemSettings( ) method.....            | 35 |
| 87  | 8.2.6  | CIM_VirtualSystemManagementService.RemoveResourceSettings( ) method .....         | 36 |
| 88  | 8.3    | Methods of the CIM_VirtualSystemSnapshotService class.....                        | 37 |
| 89  | 8.3.1  | CIM_VirtualSystemSnapshotService.CreateSnapshot( ) method .....                   | 37 |
| 90  | 8.3.2  | VirtualSystemSnapshotService.DestroySnapshot( ) method .....                      | 39 |
| 91  | 8.3.3  | VirtualSystemSnapshotService.ApplySnapshot( ) method.....                         | 40 |
| 92  | 8.4    | Profile conventions for operations .....  | 41 |
| 93  | 8.4.1  | CIM_AffectedJobElement .....  | 41 |
| 94  | 8.4.2  | CIM_ComputerSystem .....  | 42 |
| 95  | 8.4.3  | CIM_ConcreteJob .....   | 42 |
| 96  | 8.4.4  | CIM_Dependency .....  | 42 |
| 97  | 8.4.5  | CIM_ElementCapabilities .....   | 42 |
| 98  | 8.4.6  | CIM_ElementConformsToProfile .....  | 42 |
| 99  | 8.4.7  | CIM_HostedDependency.....   | 42 |
| 100 | 8.4.8  | CIM_HostedService .....   | 42 |
| 101 | 8.4.9  | CIM_LastAppliedSnapshot .....   | 42 |
| 102 | 8.4.10 | CIM_MostCurrentSnapshotInBranch.....  | 42 |
| 103 | 8.4.11 | CIM_ReferencedProfile .....   | 42 |
| 104 | 8.4.12 | CIM_RegisteredProfile.....  | 43 |
| 105 | 8.4.13 | CIM_ServiceAffectsElement .....   | 43 |
| 106 | 8.4.14 | CIM_SnapshotOfVirtualSystem .....   | 43 |
| 107 | 8.4.15 | CIM_System .....  | 43 |
| 108 | 8.4.16 | CIM_VirtualSystemManagementCapabilities .....                                     | 43 |
| 109 | 8.4.17 | CIM_VirtualSystemManagementService .....  | 43 |
| 110 | 8.4.18 | CIM_VirtualSystemSnapshotService .....  | 43 |
| 111 | 8.4.19 | CIM_VirtualSystemSnapshotCapabilities .....                                       | 43 |
| 112 | 8.4.20 | CIM_VirtualSystemSnapshotServiceCapabilities .....                                | 43 |
| 113 | 9      | Use Cases.....  | 43 |
| 114 | 9.1    | General assumptions .....   | 44 |
| 115 | 9.2    | Discovery, localization, and inspection .....                                     | 44 |
| 116 | 9.2.1  | SLP-Based discovery of CIM object managers hosting implementations of this        |    |
| 117 |        | Profile.....  | 45 |
| 118 | 9.2.2  | Locate conformant implementations using the EnumerateInstances( ) operation ..... | 46 |
| 119 | 9.2.3  | Locate conformant implementations using the ExecuteQuery( ) operation.....        | 46 |
| 120 | 9.2.4  | Locate host systems represented by central instances of this profile .....        | 46 |
| 121 | 9.2.5  | Locate implementations of scoped resource allocation profiles .....               | 47 |
| 122 | 9.2.6  | Locate virtual system management service .....                                    | 47 |
| 123 | 9.2.7  | Determine the capabilities of an implementation.....                              | 48 |
| 124 | 9.2.8  | Locate hosted resource pools of a particular resource type.....                   | 49 |
| 125 | 9.2.9  | Obtain a set of central instances of scoped resource allocation profiles.....     | 49 |
| 126 | 9.2.10 | Determine implemented resource types.....   | 50 |
| 127 | 9.2.11 | Determine the default resource pool for a resource type .....                     | 51 |
| 128 | 9.2.12 | Determine the resource pool for a resource allocation request or an allocated     |    |
| 129 |        | resource .....  | 52 |
| 130 | 9.2.13 | Determine valid settings for a resource type .....                                | 52 |
| 131 | 9.2.14 | Determine implementation class specifics.....                                     | 53 |
| 132 | 9.2.15 | Determine the implementation class for a resource type .....                      | 54 |
| 133 | 9.2.16 | Locate virtual systems hosted by a host system .....                              | 54 |
| 134 | 9.3    | Virtual system definition, modification, and destruction.....                     | 55 |
| 135 | 9.3.1  | Virtual system definition .....   | 55 |
| 136 | 9.3.2  | Virtual system modification .....   | 57 |
| 137 | 9.3.3  | Destroy virtual system .....  | 61 |
| 138 | 9.4    | Snapshot-related activities.....  | 61 |
| 139 | 9.4.1  | Locate virtual system snapshot service .....                                      | 64 |

140 9.4.2 Determine capabilities of a virtual system snapshot service ..... 64

141 9.4.3 Create snapshot..... 65

142 9.4.4 Locate snapshots of a virtual system..... 65

143 9.4.5 Locate the source virtual system of a snapshot ..... 65

144 9.4.6 Locate the most current snapshot in a branch of snapshots ..... 66

145 9.4.7 Locate dependent snapshots..... 66

146 9.4.8 Locate parent snapshot ..... 67

147 9.4.9 Apply snapshot ..... 67

148 9.4.10 Destroy snapshot..... 68

149 10 CIM elements ..... 68

150 10.1 CIM\_AffectedJobElement ..... 69

151 10.2 CIM\_ConcreteJob ..... 69

152 10.3 CIM\_Dependency ..... 70

153 10.4 CIM\_ElementCapabilities (Host system) ..... 70

154 10.5 CIM\_ElementCapabilities (Virtual system management service)..... 70

155 10.6 CIM\_ElementCapabilities (Virtual system snapshot service) ..... 71

156 10.7 CIM\_ElementCapabilities (Snapshots of virtual systems) ..... 72

157 10.8 CIM\_ElementConformsToProfile ..... 72

158 10.9 CIM\_HostedDependency..... 73

159 10.10 CIM\_HostedService (Virtual system management service) ..... 73

160 10.11 CIM\_HostedService (Virtual system snapshot service)..... 74

161 10.12 CIM\_LastAppliedSnapshot ..... 74

162 10.13 CIM\_MostCurrentSnapshotInBranch..... 75

163 10.14 CIM\_ReferencedProfile..... 75

164 10.15 CIM\_RegisteredProfile..... 76

165 10.16 CIM\_ServiceAffectsElement (Virtual system management service) ..... 76

166 10.17 CIM\_ServiceAffectsElement (Virtual system snapshot service) ..... 77

167 10.18 CIM\_SnapshotOfVirtualSystem ..... 77

168 10.19 CIM\_System ..... 78

169 10.20 CIM\_VirtualSystemManagementCapabilities ..... 78

170 10.21 CIM\_VirtualSystemManagementService ..... 79

171 10.22 CIM\_VirtualSystemSettingData (Input)..... 79

172 10.23 CIM\_VirtualSystemSettingData (Snapshot)..... 80

173 10.24 CIM\_VirtualSystemSnapshotCapabilities ..... 81

174 10.25 CIM\_VirtualSystemSnapshotService ..... 81

175 10.26 CIM\_VirtualSystemSnapshotServiceCapabilities ..... 82

176

177 **Figures**

178 Figure 1 – Profiles related to system virtualization ..... 14

179 Figure 2 – System Virtualization Profile: Class diagram..... 16

180 Figure 3 – System Virtualization Profile instance diagram: Discovery, localization, and inspection..... 45

181 Figure 4 – Virtual system configuration based on input virtual system configurations and  
182 implementation defaults ..... 56

183 Figure 5 – Virtual system resource modification ..... 60

184 Figure 6 – System Virtualization Profile: Snapshot example ..... 63

185

186 **Tables**

187 Table 1 – Related Profiles ..... 12

188 Table 2 – DefineSystem( ) method: Parameters ..... 29

189 Table 3 – DefineSystem( ) method: Return code values ..... 31

|     |  |    |
|-----|--|----|
| 190 | Table 4 – DestroySystem( ) method: Parameters .....  | 32 |
| 191 | Table 5 – DestroySystem( ) method: Return code values .....                                | 32 |
| 192 | Table 6 – AddResourceSettings( ) method: Parameters.....                                   | 33 |
| 193 | Table 7 – AddResourceSettings( ) method: Return code values .....                          | 33 |
| 194 | Table 8 – ModifyResourceSettings( ) method: Parameters .....                               | 34 |
| 195 | Table 9 – ModifyResourceSettings( ) Method: Return code values.....                        | 35 |
| 196 | Table 10 – ModifySystemSettings( ) Method: Parameters.....                                 | 36 |
| 197 | Table 11 – ModifySystemSettings( ) Method: Return code values .....                        | 36 |
| 198 | Table 12 – RemoveResourceSettings( ) Method: Parameters .....                              | 37 |
| 199 | Table 13 – RemoveResourceSettings( ) Method: Return code values .....                      | 37 |
| 200 | Table 14 – CreateSnapshot( ) method: Parameters .....                                      | 38 |
| 201 | Table 15 – CreateSnapshot( ) method: Return code values.....                               | 39 |
| 202 | Table 16 – DestroySnapshot( ) method: Parameters.....                                      | 39 |
| 203 | Table 17 – DestroySnapshot( ) method: Return code values .....                             | 40 |
| 204 | Table 18 – ApplySnapshot( ) method: Parameters .....                                       | 40 |
| 205 | Table 19 – ApplySnapshot( ) method: Return code values.....                                | 41 |
| 206 | Table 20 – CIM Elements: System Virtualization Profile .....                               | 68 |
| 207 | Table 21 – Association: CIM_AffectedJobElement .....                                       | 69 |
| 208 | Table 22 – Class: CIM_ConcreteJob .....  | 69 |
| 209 | Table 23 – Class: CIM_Dependency Class.....  | 70 |
| 210 | Table 24 – Association: CIM_ElementCapabilities (Host System).....                         | 70 |
| 211 | Table 25 – Association: CIM_ElementCapabilities (Virtual system management) .....          | 71 |
| 212 | Table 26 – Association: CIM_ElementCapabilities (Snapshot service).....                    | 71 |
| 213 | Table 27 – Association: CIM_ElementCapabilities (Snapshots of virtual systems) .....       | 72 |
| 214 | Table 28 – Association: CIM_ElementConformsToProfile.....                                  | 72 |
| 215 | Table 29 – Association: CIM_HostedDependency .....   | 73 |
| 216 | Table 30 – Association: CIM_HostedService (Virtual system management service) .....        | 73 |
| 217 | Table 31 – Association: CIM_HostedService (Virtual system snapshot service).....           | 74 |
| 218 | Table 32 – Association: CIM_LastAppliedSnapshot .....                                      | 74 |
| 219 | Table 33 – Association: CIM_MostCurrentSnapshotInBranch .....                              | 75 |
| 220 | Table 34 – Association: CIM_ReferencedProfile.....   | 75 |
| 221 | Table 35 – Class: CIM_RegisteredProfile .....  | 76 |
| 222 | Table 36 – Association: CIM_ServiceAffectsElement (Virtual system management service)..... | 77 |
| 223 | Table 37 – Association: CIM_ServiceAffectsElement .....                                    | 77 |
| 224 | Table 38 – Association: CIM_SnapshotOfVirtualSystem .....                                  | 78 |
| 225 | Table 39 – Class: CIM_VirtualSystemManagementCapabilities .....                            | 78 |
| 226 | Table 40 – Class: CIM_VirtualSystemManagementCapabilities .....                            | 78 |
| 227 | Table 41 – Class: CIM_VirtualSystemManagementService .....                                 | 79 |
| 228 | Table 42 – Class: CIM_VirtualSystemSettingData (Input) .....                               | 80 |
| 229 | Table 43 – Class: CIM_VirtualSystemSettingData (Snapshot) .....                            | 80 |
| 230 | Table 44 – Class: CIM_VirtualSystemSnapshotCapabilities.....                               | 81 |
| 231 | Table 45 – Class: CIM_VirtualSystemSnapshotService .....                                   | 81 |
| 232 | Table 46 – Class: CIM_VirtualSystemSnapshotServiceCapabilities.....                        | 82 |
| 233 |  |    |

234

## Foreword

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

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

### 239 Acknowledgments

240 The authors wish to acknowledge the following people.

241 Editor:

- 242 • Michael Johanssen – IBM

243 Contributors:

- 244 • Gareth Bestor – IBM
- 245 • Chris Brown – HP
- 246 • Mike Dutch – Symantec
- 247 • Jim Fehlig – Novell
- 248 • Kevin Fox – Sun Microsystems, Inc.
- 249 • Ron Goering – IBM
- 250 • Daniel Hiltgen – EMC/VMware
- 251 • Michael Johanssen – IBM
- 252 • Larry Lamers – EMC/VMware
- 253 • Andreas Maier – IBM
- 254 • Aaron Merkin – IBM
- 255 • John Parchem – Microsoft
- 256 • Nihar Shah – Microsoft
- 257 • David Simpson – IBM
- 258 • Carl Waldspurger – EMC/VMware

259

## Introduction

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



## 266 System Virtualization Profile

### 267 1 Scope

268 This profile is an autonomous profile that specifies the minimum top-level object model needed for the  
269 representation of host systems and the discovery of hosted virtual computer systems. In addition, it  
270 specifies a service for the manipulation of virtual computer systems and their resources, including  
271 operations for the creation, deletion, and modification of virtual computer systems and operations for the  
272 addition or removal of virtual resources to or from virtual computer systems.

### 273 2 Normative references

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

277 DMTF DSP0004, *CIM Infrastructure Specification 2.5*  
278 [http://www.dmtf.org/standards/published\\_documents/DSP0004\\_2.5.pdf](http://www.dmtf.org/standards/published_documents/DSP0004_2.5.pdf)

279 DMTF DSP0200, *CIM Operations over HTTP 1.3*  
280 [http://www.dmtf.org/standards/published\\_documents/DSP0200\\_1.3.pdf](http://www.dmtf.org/standards/published_documents/DSP0200_1.3.pdf)

281 DMTF DSP0201, *Representation of CIM in XML 2.3*  
282 [http://www.dmtf.org/standards/published\\_documents/DSP0201\\_2.3.pdf](http://www.dmtf.org/standards/published_documents/DSP0201_2.3.pdf)

283 DMTF DSP1001, *Management Profile Specification Usage Guide 1.0*  
284 [http://www.dmtf.org/standards/published\\_documents/DSP1001\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1001_1.0.pdf)

285 DMTF DSP1012, *Boot Control Profile 1.0*  
286 [http://www.dmtf.org/standards/published\\_documents/DSP1012\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1012_1.0.pdf)

287 DMTF DSP1022, *CPU Profile 1.0*  
288 [http://www.dmtf.org/standards/published\\_documents/DSP1022\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1022_1.0.pdf)

289 DMTF DSP1027, *Power State Management Profile 1.0*  
290 [http://www.dmtf.org/standards/published\\_documents/DSP1027\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1027_1.0.pdf)

291 DMTF DSP1033, *Profile Registration Profile 1.0*  
292 [http://www.dmtf.org/standards/published\\_documents/DSP1033\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1033_1.0.pdf)

293 DMTF DSP1041, *Resource Allocation Profile 1.1*  
294 [http://www.dmtf.org/standards/published\\_documents/DSP1041\\_1.1.pdf](http://www.dmtf.org/standards/published_documents/DSP1041_1.1.pdf)

295 DMTF DSP1043, *Allocation Capabilities Profile 1.0*  
296 [http://www.dmtf.org/standards/published\\_documents/DSP1043\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1043_1.0.pdf)

297 DMTF DSP1044, *Processor Device Resource Virtualization Profile 1.0*  
298 [http://www.dmtf.org/standards/published\\_documents/DSP1044\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1044_1.0.pdf)

299 DMTF DSP1045, *Memory Resource Virtualization Profile 1.0*  
300 [http://www.dmtf.org/standards/published\\_documents/DSP1045\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1045_1.0.pdf)

301 DMTF DSP1047, *Storage Resource Virtualization Profile 1.0*  
302 [http://www.dmtf.org/standards/published\\_documents/DSP1047\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1047_1.0.pdf)

- 303 DMTF DSP1052, *Computer System Profile 1.0*  
304 [http://www.dmtf.org/standards/published\\_documents/DSP1052\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1052_1.0.pdf)
- 305 DMTF DSP1053, *Base Metrics Profile 1.0*  
306 [http://www.dmtf.org/standards/published\\_documents/DSP1053\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1053_1.0.pdf)
- 307 DMTF DSP1057, *Virtual System Profile 1.0*  
308 [http://www.dmtf.org/standards/published\\_documents/DSP1057\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1057_1.0.pdf)
- 309 DMTF DSP1059, *Generic Device Resource Virtualization Profile 1.0*  
310 [http://www.dmtf.org/standards/published\\_documents/DSP1059\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1059_1.0.pdf)
- 311 ISO/IEC Directives, Part2:2004, *Rules for the structure and drafting of International Standards*,  
312 <http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype>

### 313 **3 Terms and definitions**

314 For the purposes of this document, the following terms and definitions apply. For the purposes of this  
315 document, the terms and definitions in [DSP1033](#) and [DSP1001](#) also apply.

#### 316 **3.1**

##### 317 **can**

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

#### 319 **3.2**

##### 320 **cannot**

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

#### 322 **3.3**

##### 323 **conditional**

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

#### 326 **3.4**

##### 327 **mandatory**

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

#### 330 **3.5**

##### 331 **may**

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

#### 333 **3.6**

##### 334 **need not**

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

#### 336 **3.7**

##### 337 **optional**

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

#### 339 **3.8**

##### 340 **referencing profile**

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

- 343 **3.9**  
344 **shall**  
345 indicates requirements to be followed strictly in order to conform to the document and from which no  
346 deviation is permitted
- 347 **3.10**  
348 **shall not**  
349 indicates requirements to be followed strictly in order to conform to the document and from which no  
350 deviation is permitted
- 351 **3.11**  
352 **should**  
353 indicates that among several possibilities, one is recommended as particularly suitable, without mention-  
354 ing or excluding others, or that a certain course of action is preferred but not necessarily required
- 355 **3.12**  
356 **should not**  
357 indicates that a certain possibility or course of action is deprecated but not prohibited
- 358 **3.13**  
359 **unspecified**  
360 indicates that this profile does not define any constraints for the referenced CIM element
- 361 **3.14**  
362 **implementation**  
363 a set of software components that realize the classes that are specified or specialized by this profile
- 364 **3.15**  
365 **client**  
366 application that exploits facilities specified by this profile
- 367 **3.16**  
368 **this profile**  
369 a reference to this DMTF management profile: DSP1042 (*System Virtualization Profile*)
- 370 **3.17**  
371 **virtualization platform**  
372 virtualizing infrastructure provided by a host system that enables the deployment of virtual systems
- 373 **3.18**  
374 **WBEM service**  
375 a component that provides a service accessible through a WBEM protocol  
376 A single WBEM service instance may be used by multiple WBEM client instances. The term WBEM  
377 service is used to denote the entire set of components on the server side that is needed to provide the  
378 service. For example, in typical WBEM infrastructures this includes a CIM object manager and a set of  
379 CIM providers.

## 380 **4 Symbols and abbreviated terms**

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

- 382 **4.1**  
383 **RASD**  
384 resource allocation setting data

385 **4.2**  
 386 **SLP**  
 387 service location protocol

388 **4.3**  
 389 **VS**  
 390 virtual system

391 **4.4**  
 392 **VSSD**  
 393 virtual system setting data

## 394 **5 Synopsis**

395 **Profile Name:** System Virtualization

396 **Version:** 1.0.0

397 **Organization:** DMTF

398 **CIM Schema Version:** 2.22

399 **Central Class:** CIM\_System

400 **Scoping Class:** CIM\_System

401 This profile is an autonomous profile that defines the minimum object model for the representation of host  
 402 systems. It identifies component profiles that address the allocation of resources. It extends the object  
 403 model for the representation of virtual systems and virtual resources defined in [DSP1057](#).

404 The central instance and the scoping instance of this profile shall be an instance of the CIM\_System class  
 405 that represents a host system.

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

408

**Table 1 – Related Profiles**

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

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

## 409 6 Description

410 This clause contains informative text only.

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

- 413 • the detection of host systems that provide system virtualization facilities
- 414 • the discovery of scoped host resources
- 415 • the discovery of scoped resource pools
- 416 • the inspection of host system capabilities for
  - 417 – the creation and manipulation of virtual systems
  - 418 – the allocation of resources of various types
- 419 • the inspection of resource pool capabilities
- 420 • the discovery of hosted virtual systems
- 421 • the inspection of relationships between host entities (host systems, host resources, and re-  
422 source pools) and virtual entities (virtual systems and virtual resources)
- 423 • the creation and manipulation of virtual systems using input configurations, predefined  
424 configurations available at the host system, or both
- 425 • the creation and manipulation of snapshots that capture the configuration and state of a virtual  
426 system at a particular point in time

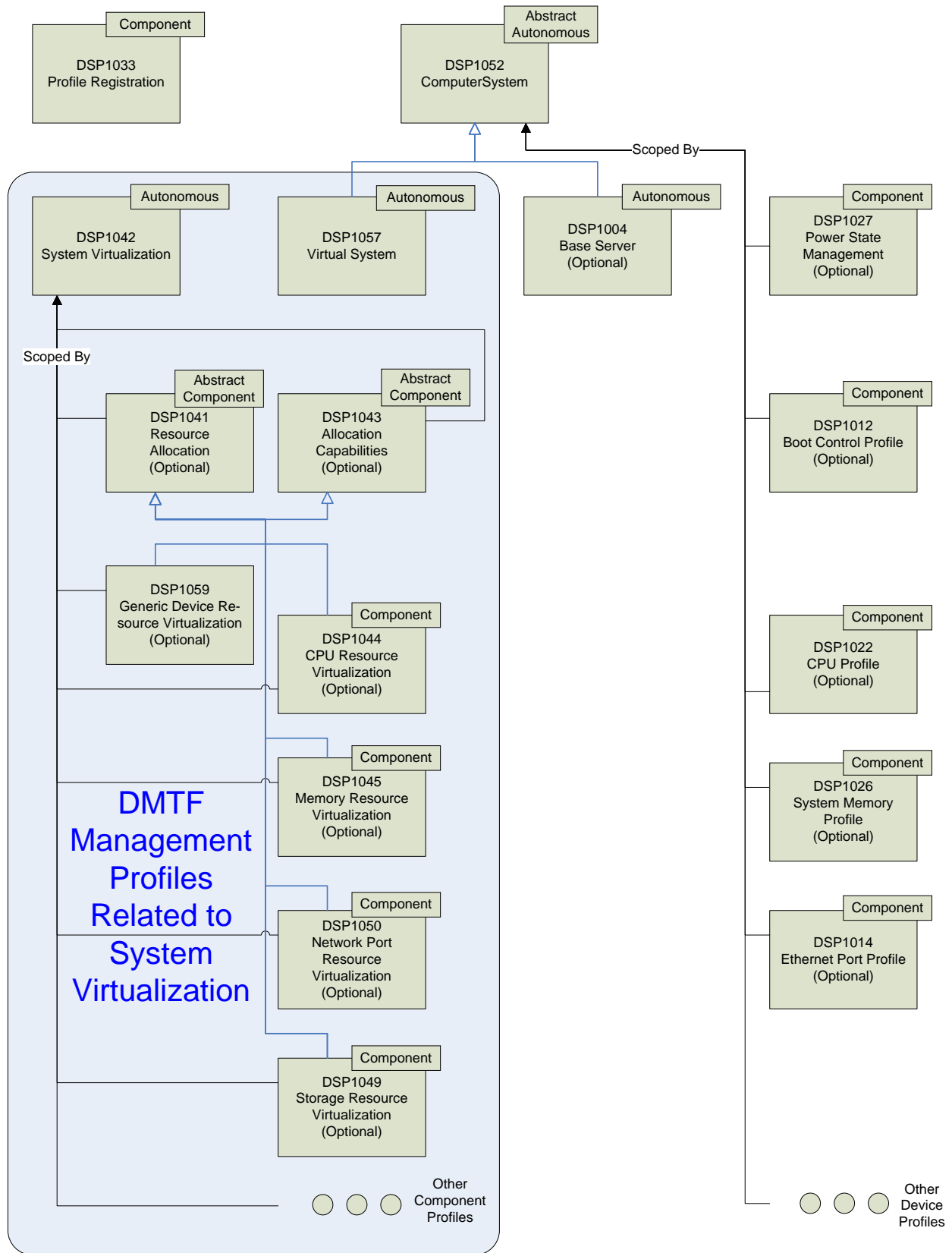
### 427 6.1 Profile relationships

428 A client that is exploiting system virtualization facilities specified by this profile needs to be virtualization  
429 aware. The specified model keeps that knowledge at an abstract level that is independent of a particular  
430 system virtualization platform implementation or technology.

431 This profile complements [DSP1057](#).

- 432 • This profile focuses on virtualization aspects related to host systems and their resources, such  
433 as modeling the relationships between host resources and virtual resources. Further it  
434 addresses virtualization-specific tasks such as the creation or modification of virtual systems  
435 and their configurations.
- 436 • [DSP1057](#) defines a top-level object model for the inspection and basic operation of virtual  
437 systems. It is a specialization of [DSP1052](#) that defines a management interface for general-  
438 purpose computer systems. Consequently, the interface specified for the basic inspection and  
439 operation of virtual systems is conformant with that specified for real systems. A client that is  
440 exploiting capabilities specified by [DSP1052](#) with respect to virtual systems that are instrument  
441 conformant with [DSP1057](#) can inherently handle virtual systems like real systems without being  
442 virtualization aware.

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



444

445

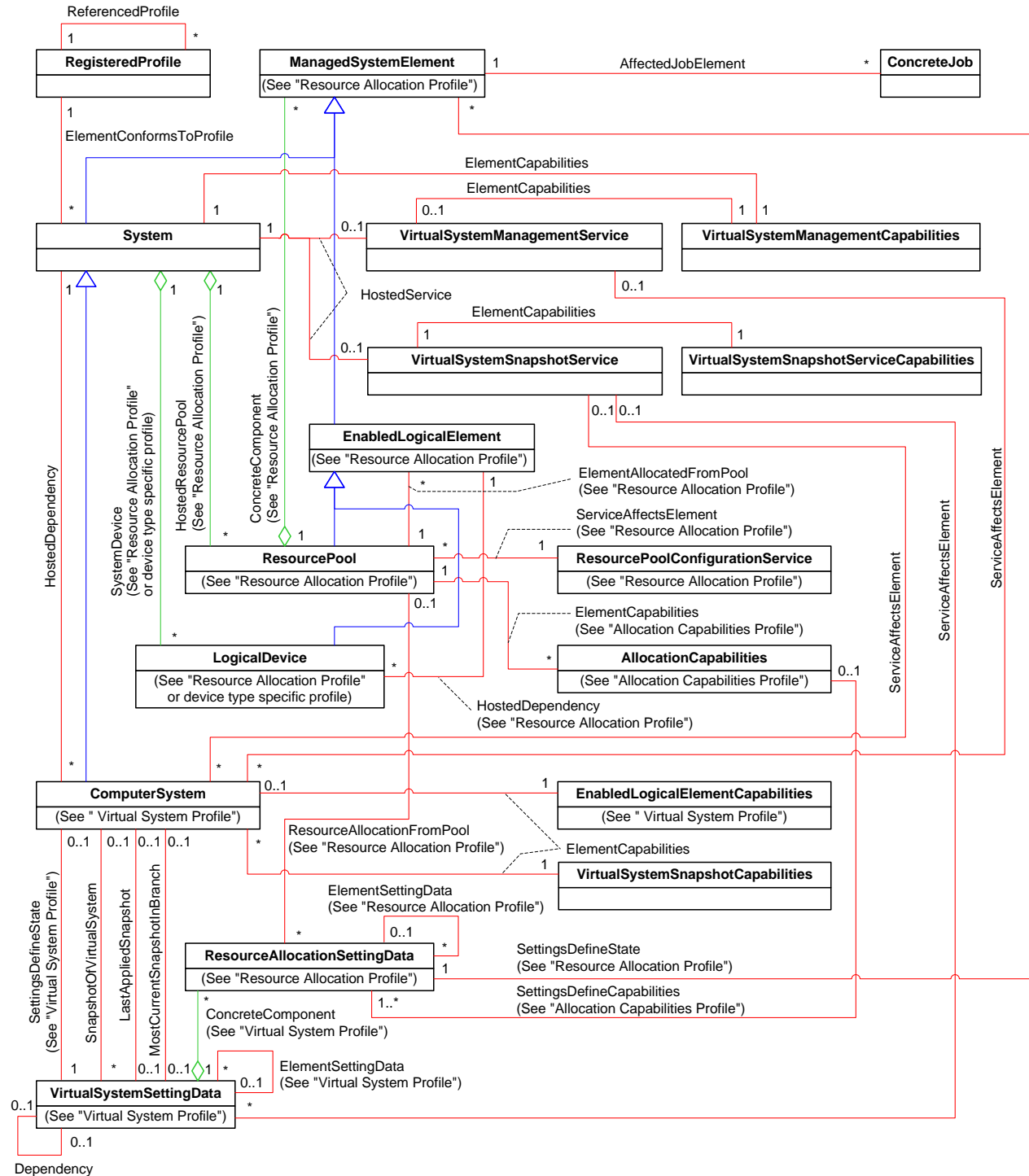
Figure 1 – Profiles related to system virtualization

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

- 448 • This profile
- 449 This profile enables the inspection of host systems, their resources, their capabilities, and their  
 450 services for creation and manipulation of virtual systems.
- 451 • [DSP1057](#)
- 452 [DSP1057](#) enables the inspection of and basic operations on virtual systems.
- 453 • Resource-type-specific profiles
- 454 Resource-type-specific profiles enable the inspection and operation of resources for one  
 455 particular resource type. They apply to both virtual and host resources; they do not cover  
 456 virtualization-specific aspects of resources. A client may exploit resource-type-specific profiles  
 457 for the inspection and manipulation of virtual and host resources in a similar manner.
- 458 • Resource allocation profiles
- 459 Resource allocation profiles enable the inspection and management of resource allocation re-  
 460 quests, allocated resources, and resources available for allocation. Resource allocation profiles  
 461 are based on [DSP1041](#) and on [DSP1043](#). Resource allocation profiles are scoped by this  
 462 profile. A client may exploit resource allocation profiles for the inspection of
  - 463 – allocated resources
  - 464 – allocation dependencies that virtual resources have on host resources and resource pools
  - 465 – capabilities that describe possible values for allocation requests
  - 466 – capabilities that describe the mutability of resource allocations
- 467 For some resource types, specific resource allocation profiles are specified that address re-  
 468 source-type-specific resource allocation aspects and capabilities. Examples are [DSP1044](#) and  
 469 [DSP1047](#).
- 470 The management of the allocation of basic virtual resources that are not covered by a resource-  
 471 type-specific resource allocation profile is specified in [DSP1059](#).

## 472 6.2 System virtualization class schema

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



477  
478

479 **Figure 2 – System Virtualization Profile: Class diagram**

480 This profile specifies the use of the following classes and associations:

- 481 • the CIM\_RegisteredProfile class and the CIM\_ElementConformsToProfile association for the
- 482 advertisement of conformance to this profile



- 483 • the CIM\_ReferencedProfile association for the representation of a scoping relationship between  
484 this profile and scoped DMTF management profiles
- 485 • the CIM\_System class for the representation of host systems
- 486 • the CIM\_HostedDependency association for the representation of the hosting relationship be-  
487 tween a host system and hosted virtual systems
- 488 • the CIM\_VirtualSystemManagementService class for the representation of virtual system  
489 management services available at a host system, providing operations like the creation and  
490 modification of virtual systems and their components
- 491 • the CIM\_HostedService association for the representation of the relationship between a host  
492 system and services that it provides
- 493 • the CIM\_VirtualSystemManagementCapabilities class for the representation of optional fea-  
494 tures, properties, and methods available for the management of virtual systems hosted by a  
495 host system
- 496 • the CIM\_ElementCapabilities association for the representation of the relationship between a  
497 host system, a virtual system or a service, and their respective capabilities
- 498 • the CIM\_ServiceAffectsElement association for the representation of the relationship between  
499 defined services and affected elements like virtual systems or virtual system snapshots
- 500 • the CIM\_VirtualSystemSettingData class for the representation of snapshots (in addition to the  
501 use of that class for the representation of virtual aspects of a virtual system as specified by  
502 [DSP1057](#))
- 503 • the CIM\_VirtualSystemSnapshotService class for the representation of snapshot-related ser-  
504 vices available at a host system
- 505 • the CIM\_VirtualSystemSnapshotServiceCapabilities class for the representation of optional fea-  
506 tures, properties, and methods available for the management of snapshots of virtual systems
- 507 • the CIM\_VirtualSystemSnapshotCapabilities class for the representation of optional features,  
508 properties, and methods available for the management of snapshots relating to one particular  
509 virtual system
- 510 • the CIM\_SnapshotOfVirtualSystem association for the representation of the relationship be-  
511 tween a snapshot of a virtual system and the virtual system itself
- 512 • the CIM\_Dependency association for dependencies among virtual system snapshots
- 513 • the CIM\_LastAppliedSnapshot association for the representation of the relationship between a  
514 virtual system and the snapshot that was most recently applied to it
- 515 • the CIM\_MostCurrentSnapshotInBranch association for the representation of the relationship  
516 between a virtual system and the snapshot that is the most current snapshot in a sequence of  
517 snapshots captured from the virtual system
- 518 • the CIM\_ConcreteJob class and the CIM\_AffectedJobElement association to model a mecha-  
519 nism that allows tracking of asynchronous tasks resulting from operations such as the optional  
520 CreateSystem( ) method of the CIM\_VirtualSystemManagementService class

521 In general, any mention of a class in this document means the class itself or its subclasses. For example,  
522 a statement such as "an instance of the CIM\_LogicalDevice class" implies an instance of the CIM\_Logi-  
523 calDevice class or a subclass of the CIM\_LogicalDevice class.

### 524 **6.3 Virtual system configurations**

525 This profile extends the use of virtual system configurations. [DSP1057](#) defines a virtual system  
526 configuration as one top-level instance of the CIM\_VirtualSystemSettingData class that aggregates zero

527 or more instances of the CIM\_ResourceAllocationSettingData class through the CIM\_VirtualSystemSet-  
528 tingDataComponent association.

529 [DSP1057](#) defines the concept of virtual system configurations and applies it to the following types of  
530 virtual system configurations:

- 531 • the "State" virtual system configuration, which represents a virtualization-specific state that ex-  
532 tends a virtual system representation
- 533 • the "Defined" virtual system configuration, which represents virtual system definitions
- 534 • the "Next" virtual system configuration, which represents the virtual system configuration that  
535 will be used for the next activation of a virtual system

536 This profile applies the concept of virtual system configurations and defines the following additional types  
537 of virtual system configurations:

- 538 • the "Input" virtual system configuration, which represents configuration information for new vir-  
539 tual systems
- 540 • the "Reference" virtual system configuration, which represents configuration information that  
541 complements an "Input" virtual system configuration for a new virtual system
- 542 • the "Snapshot" virtual system configuration, which represents snapshots of virtual systems

#### 543 **6.4 Resource allocation**

544 An allocated resource is a resource subset or resource share that is allocated from a resource pool. An  
545 allocated resource is obtained based on a resource allocation request. Both allocated resources and  
546 resource allocation requests are represented through instances of the  
547 CIM\_ResourceAllocationSettingData class.

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

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

- 552 • Persistent Resource Allocation

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

- 555 • Transient Resource Allocation

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

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

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

562 EXAMPLE 2: Transient Resource Allocation: Host memory

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

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

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

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

## 574 **6.5 Snapshots**

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

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

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

587 This profile specifies mechanisms for the creation, application, and destruction of snapshots. It specifies a  
588 snapshot model that enables the inspection of snapshot-related configuration information such as the  
589 virtual system configurations that were effective when the snapshot was captured. Relationships between  
590 snapshots are also modeled.

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

## 594 **7 Implementation**

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

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

599 Where reference is made to CIM Schema properties that enumerate values, the numeric value is norma-  
600 tive and the descriptive text following it in parentheses is informational. For example, in the statement "If  
601 an instance of the CIM\_VirtualSystemManagementCapabilities class contains the value  
602 3 (DestroySystemSupported) in an element of the SynchronousMethodsSupported[ ] array property," the  
603 value "3" is normative text and "(DestroySystemSupported)" is informational text.

### 604 **7.1 Host system**

605 The CIM\_System class shall be used for the representation of host systems. There shall be one instance  
606 of the CIM\_System class for each host system that is managed conformant to this profile.

### 607 **7.2 Profile registration**

608 [DSP1033](#) describes how an implementation of a profile shall advertise that a profile is implemented.

### 609 7.2.1 This profile

610 The implementation of this profile shall be indicated by an instance of the CIM\_RegisteredProfile class in  
611 the CIM Interop namespace. Each instance of the CIM\_System class that represents a host system that  
612 is manageable through this profile shall be a central instance of this profile by associating it with the  
613 instance of the CIM\_RegisteredProfile class through an instance of the CIM\_ElementConformsToProfile  
614 association.

### 615 7.2.2 Scoped resource allocation profiles

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

618 The support of scoped resource-allocation profiles is conditional with respect to the presence of an in-  
619 stance of the CIM\_RegisteredProfile class in the Interop namespace that represents the scoped resour-  
620 ce-allocation profile implementation and is associated with the instance of the CIM\_RegisteredProfile  
621 class that represents an implementation of this profile through an instance of the CIM\_ReferencedProfile  
622 association.

623 Resource-allocation DMTF management profiles are based on [DSP1041](#) and [DSP1043](#). The resource-  
624 allocation DMTF management profiles that are scoped by this profile are listed in Table 1, starting with  
625 [DSP1044](#).

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

- 628 • If a resource-type-specific resource-allocation DMTF management profile is specified for that re-  
629 source type, that profile should be implemented.
- 630 • If no resource-type-specific resource-allocation DMTF management profile exists at version 1.0  
631 or later, [DSP1059](#) should be implemented.

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

- 634 • The instance of the CIM\_RegisteredProfile class that represents the implementation of this pro-  
635 file and the instance of the CIM\_RegisteredProfile class that represents the implementation of  
636 the scoped resource-allocation DMTF management profile shall be associated through an in-  
637 stance of the CIM\_ReferencedProfile association.
- 638 • One of the following conditions regarding profile implementation advertisement shall be met:
  - 639 – Central Class Profile Implementation Advertisement:  
640 Instances of the CIM\_ElementConformsToProfile association shall associate each instance  
641 of the CIM\_ResourcePool class that is a central instance of the scoped-resource-allocation  
642 DMTF management profile with the instance of the CIM\_RegisteredProfile class that repre-  
643 sents an implementation of the scoped-resource-allocation DMTF management profile.
  - 644 – Scoping Class Profile Implementation Advertisement:  
645 No instances of the CIM\_ElementConformsToProfile association shall associate any in-  
646 stance of the CIM\_ResourcePool class that is a central instance of the scoped-resource-  
647 allocation DMTF management profile with the instance of the CIM\_RegisteredProfile class  
648 that represents an implementation of the scoped-resource-allocation DMTF management  
649 profile.

## 650 7.3 Representation of hosted virtual systems

651 This profile strengthens the requirements for the representation of virtual system configurations specified  
652 by [DSP1057](#) for hosted virtual systems.

### 653 **7.3.1 Profile conformance for hosted virtual systems**

654 Any virtual system that is hosted by a conformant host system shall be represented by an instance of the  
655 CIM\_ComputerSystem class that is a central instance of [DSP1057](#). That instance shall be associated with  
656 the instance of the CIM\_System class that represents the conformant host system through an instance of  
657 the CIM\_HostedDependency association.

### 658 **7.3.2 CIM\_VirtualSystemSettingData.VirtualSystemType property**

659 The value of the VirtualSystemType property shall be equal to an element of the  
660 VirtualSystemTypesSupported[ ] array property in the instance of the  
661 CIM\_VirtualSystemManagementCapabilities class that is associated with the instance of the  
662 CIM\_VirtualSystemManagementService class that represents the host system, or shall be NULL if the  
663 value of the VirtualSystemTypesSupported[ ] array property is NULL (see 7.4.2).

## 664 **7.4 Virtual system management capabilities**

665 This subclause models capabilities of virtual system management in terms of the  
666 CIM\_VirtualSystemManagementCapabilities class.

### 667 **7.4.1 CIM\_VirtualSystemManagementCapabilities class**

668 An instance of the CIM\_VirtualSystemManagementCapabilities class shall be used to represent the virtual  
669 system management capabilities of a host system. That instance shall be associated with the instance of  
670 the CIM\_System class that represents the host system through the CIM\_ElementCapabilities association.

### 671 **7.4.2 CIM\_VirtualSystemManagementCapabilities.VirtualSystemTypesSupported[ ] array 672 property**

673 The implementation of the VirtualSystemTypesSupported[ ] array property is optional. The  
674 VirtualSystemTypesSupported[ ] array property should be implemented.

675 If the VirtualSystemTypesSupported[ ] array property is implemented, the provisions in this subclause  
676 apply.

677 Array values shall designate the set of supported virtual system types. If the  
678 VirtualSystemTypesSupported[ ] array property is not implemented (has a value of NULL), the  
679 implementation does not externalize the set of implemented virtual system types, but internally still may  
680 exhibit different types of virtual systems.

### 681 **7.4.3 CIM\_VirtualSystemManagementCapabilities.SynchronousMethodsSupported[ ] 682 array property**

683 The implementation of the SynchronousMethodsSupported[ ] array property is optional. The  
684 SynchronousMethodsSupported[ ] array property should be implemented.

685 If the SynchronousMethodsSupported[ ] array property is implemented, the provisions in this subclause  
686 apply.

687 Array values shall designate the set of methods of the CIM\_VirtualSystemManagementService class that  
688 are implemented with synchronous behavior only. A NULL value or an empty value set shall be used to  
689 indicate that no methods are implemented with synchronous behavior. If a method is designated within  
690 the value set of the SynchronousMethodsSupported[ ] property, that method shall always exhibit  
691 synchronous behavior and shall not be designated within the value set of the  
692 AsynchronousMethodsSupported[ ] property.

693 **7.4.4 CIM\_VirtualSystemManagementCapabilities.AsynchronousMethodsSupported[ ]**  
694 **array property**

695 The implementation of the AsynchronousMethodsSupported[ ] array property is optional. The  
696 AsynchronousMethodsSupported[ ] array property should be implemented.

697 If the AsynchronousMethodsSupported[ ] array property is implemented, the provisions in this subclause  
698 apply.

699 Array values shall designate the set of methods of the CIM\_VirtualSystemManagementService class that  
700 are implemented with synchronous and potentially with asynchronous behavior. A NULL value or an  
701 empty value set shall be used to indicate that no methods are implemented with asynchronous behavior.  
702 If a method is designated with a value in the AsynchronousMethodsSupported[ ] array property, it may  
703 show either synchronous or asynchronous behavior.

704 **7.4.5 CIM\_VirtualSystemManagementCapabilities.IndicationsSupported[ ] array**  
705 **property**

706 The implementation of the IndicationsSupported[ ] array property is optional. The IndicationsSupported[ ]  
707 array property should be implemented.

708 If the IndicationsSupported[ ] array property is implemented, the provisions in this subclause apply.

709 Array values shall designate the set of types of indications that are implemented. A NULL value or an  
710 empty value set shall be used to indicate that indications are not implemented.

711 **7.4.6 Grouping Rules for implementations of methods of the**  
712 **CIM\_VirtualSystemManagementService class**

713 The grouping rules specified in this subclause shall be applied for implementations of methods of the  
714 CIM\_VirtualSystemManagementService class. Within a group either all methods or no method at all shall  
715 be implemented; nevertheless synchronous and asynchronous behavior may be mixed.

716 **7.4.6.1 Virtual system definition and destruction**

717 If virtual system definition and destruction are implemented, the DefineSystem( ) and DestroySystem( )  
718 methods of the CIM\_VirtualSystemManagementService class shall be implemented, and the values  
719 2 (DefineSystemSupported) and 3 (DestroySystemSupported) shall be set in the  
720 SynchronousMethodsSupported[ ] or AsynchronousMethodsSupported[ ] array properties within the  
721 instance of the CIM\_VirtualSystemManagementCapabilities class that describes capabilities of the imple-  
722 mentation.

723 If virtual system definition and destruction are not implemented, the values 2 (DefineSystemSupported)  
724 and 3 (DestroySystemSupported) shall not be set in the SynchronousMethodsSupported[ ] or  
725 AsynchronousMethodsSupported[ ] array properties of the instance of the  
726 CIM\_VirtualSystemManagementCapabilities class that describes the virtual system management capabili-  
727 ties of the host system.

728 **7.4.6.2 Virtual resource addition and removal**

729 If the addition and removal of virtual resources to or from virtual systems are implemented, the  
730 AddResourceSettings( ) and RemoveResourceSettings( ) methods of the  
731 CIM\_VirtualSystemManagementService class shall be implemented, and the values  
732 1 (AddResourceSettingsSupported) and 7 (RemoveResourceSettingsSupported) shall be set in the  
733 SynchronousMethodsSupported[ ] or AsynchronousMethodsSupported[ ] array properties of the instance  
734 of the CIM\_VirtualSystemManagementCapabilities class that describes the virtual system management  
735 capabilities of the host system.

736 If the addition and removal of virtual resources to virtual systems is not implemented, the values  
737 1 (AddResourceSettingsSupported) and 7 (RemoveResourceSettingsSupported) shall not be set in the  
738 SynchronousMethodsSupported[ ] or AsynchronousMethodsSupported[ ] array properties of the instance  
739 of the CIM\_VirtualSystemManagementCapabilities class that describes the virtual system management  
740 capabilities of the host system.

#### 741 **7.4.6.3 Virtual system and resource modification**

742 If the modification of virtual systems and virtual resources is implemented, the ModifyResourceSettings( )  
743 and ModifySystemSettings( ) methods of the CIM\_VirtualSystemManagementService class shall be  
744 implemented, and the values 5 (ModifyResourceSettingsSupported) and  
745 6 (ModifySystemSettingsSupported) shall be set in the SynchronousMethodsSupported[ ] or  
746 AsynchronousMethodsSupported[ ] array properties of the instance of the  
747 CIM\_VirtualSystemManagementCapabilities class that describes the virtual system management capabili-  
748 ties of the host system.

749 If the modification of virtual systems and virtual resources is not implemented, the values  
750 5 (ModifyResourceSettingsSupported) and 6 (ModifySystemSettingsSupported) shall not be set in the  
751 SynchronousMethodsSupported[ ] or AsynchronousMethodsSupported[ ] array properties of the instance  
752 of the CIM\_VirtualSystemManagementCapabilities class that describes the virtual system management  
753 capabilities of the host system.

### 754 **7.5 Virtual system definition and modification**

755 This profile specifies methods for the definition and modification of virtual systems. These method  
756 specifications use the CIM\_VirtualSystemSettingData class for the parameterization of system-specific  
757 properties. Subsequent subclauses specify:

- 758 • how a client shall prepare instances of the CIM\_VirtualSystemSettingData class that are used  
759 as a parameter for a method that defines or modifies a virtual system
- 760 • how an implementation shall interpret instances of the CIM\_VirtualSystemSettingData class that  
761 are used as a parameter for a method that defines or modifies a virtual system

762 Definition requests for virtual systems are modeled through the  
763 CIM\_VirtualSystemManagementService.DefineSystem( ) method, and modification requests for virtual  
764 system properties are modeled through the  
765 CIM\_VirtualSystemManagementService.ModifySystemSettings( ) method.

#### 766 **7.5.1 CIM\_VirtualSystemSettingData.InstanceID property**

767 A client shall set the value of the InstanceID property to NULL if the instance of the  
768 CIM\_VirtualSystemSettingData class is created locally. A client shall not modify the value of the  
769 InstanceID property in an instance of the CIM\_VirtualSystemSettingData class that was received from an  
770 implementation and is sent back to the implementation as a parameter of a modification method.

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

773 An implementation shall use a non-NULL value to identify an existing instance of the  
774 CIM\_VirtualSystemSettingData class. If the value does not identify an instance of the  
775 CIM\_VirtualSystemSettingData class, an implementation shall return a return code that indicates an inval-  
776 id parameter (see 8.2.4.3).

#### 777 **7.5.2 CIM\_VirtualSystemSettingData.ElementName property**

778 The implementation of the ElementName property is optional.



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

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

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

786 If the implementation supports modification requests that affect the value of the ElementName property,  
787 the implementation shall support the CIM\_EnabledLogicalElementCapabilities class for virtual systems as  
788 specified in [DSP1052](#).

### 789 **7.5.3 CIM\_VirtualSystemSettingData.VirtualSystemIdentifier property**

790 The implementation of the VirtualSystemIdentifier property is optional.

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

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

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

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

### 802 **7.5.4 CIM\_VirtualSystemSettingData.VirtualSystemType property**

803 The implementation of the VirtualSystemType property is optional.

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

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

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

## 816 **7.6 Virtual resource definition and modification**

817 This profile specifies how to define and modify virtual resources using methods of the virtual system  
818 management service. In these method specifications, the CIM\_ResourceAllocationSettingData class is  
819 used for parameterization of resource allocation specific properties. For specifications that define the use  
820 of the CIM\_ResourceAllocationSettingData class, see [DSP1041](#), [DSP1043](#), and profiles that specialize  
821 these (for example, [DSP1059](#)). [DSP1041](#) describes the use of the CIM\_ResourceAllocationSettingData



822 class, and [DSP1043](#) introduces the concept of allowing a client to determine the acceptable value sets for  
 823 values of properties of the CIM\_ResourceAllocationSettingData class in virtual resource definition and  
 824 modification requests.

## 825 **7.7 Virtual system snapshots**

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

827 The implementation of virtual system snapshots is optional.

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

### 829 **7.7.1 Virtual system snapshot service and capabilities**

830 This subclause models elements of virtual system snapshot management in terms of the  
 831 CIM\_VirtualSystemSnapshotService class and the CIM\_VirtualSystemSnapshotServiceCapabilities class.

#### 832 **7.7.1.1 Virtual system snapshots**

833 The implementation of virtual system snapshots is optional.

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

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

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

- 837 • the CIM\_VirtualSystemSnapshotService class shall be implemented and the following methods  
 838 shall be implemented:
  - 839 – CreateSnapshot( ), for at least one type of snapshot
  - 840 – DestroySnapshot( )
  - 841 – ApplySnapshot( )
- 842 • There shall be exactly one instance of the CIM\_VirtualSystemSnapshotService class associated  
 843 to the central instance of this profile through an instance of the CIM\_HostedService association.

844 If virtual system snapshots are not implemented, the CIM\_VirtualSystemSnapshotService class shall not  
 845 be implemented.

#### 846 **7.7.1.2 CIM\_VirtualSystemSnapshotServiceCapabilities class**

847 The provisions in this subclause are conditional.

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

849 If the CIM\_VirtualSystemSnapshotServiceCapabilities class is implemented, the provisions in this  
 850 subclause apply.

851 An instance of the CIM\_VirtualSystemSnapshotServiceCapabilities class shall be used to represent the  
 852 capabilities of the virtual system snapshot service of a host system. The instance shall be associated with  
 853 the instance of the CIM\_VirtualSystemSnapshotService class that represents the virtual system snapshot  
 854 service through the CIM\_ElementCapabilities association.

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

- 858 • 2 (CreateSnapshotSupported)
- 859 • 3 (DestroySnapshotSupported)

- 4 (ApplySnapshotSupported)

861 The implementation of the SynchronousMethodsSupported[ ] array property is conditional with respect to  
862 at least one of the snapshot methods being implemented with synchronous behavior. A NULL value or an  
863 empty value set shall be used to indicate that no methods are implemented with synchronous behavior. If  
864 a method is designated within the value set of the SynchronousMethodsSupported[ ] property, that  
865 method shall always exhibit synchronous behavior and shall not be designated within the value set of the  
866 AsynchronousMethodsSupported[ ] property.

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

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

## 873 **7.7.2 Virtual system snapshot representation**

874 The provisions in this subclause are conditional.

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

876 If the representation of virtual system snapshots is implemented, the provisions in this subclause apply.

877 Snapshots of virtual systems shall be represented by instances of the CIM\_VirtualSystemSettingData  
878 class. Each such instance shall be associated with the instance of the CIM\_ComputerSystem class that  
879 represents the virtual system that was the source of the snapshot through an instance of the  
880 CIM\_SnapshotOfVirtualSystem association.

## 881 **7.7.3 Designation of the last applied snapshot**

882 The provisions in this subclause are conditional.

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

884 If the designation of the last applied snapshot is implemented, the provisions in this subclause apply.

885 If a snapshot was applied to a virtual system, an instance of the CIM\_LastAppliedSnapshot association  
886 shall connect the instance of the CIM\_ComputerSystem class that represents the virtual system and the  
887 instance of the CIM\_VirtualSystemSettingData class that represents the snapshot. The association  
888 instance shall be actualized as different snapshots are applied.

## 889 **7.7.4 Designation of the most current snapshot in branch**

890 The implementation of the representation the most current snapshot in a branch is conditional.

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

892 If the designation of the most current snapshot in a branch is implemented, the provisions in this  
893 subclause apply.

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

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

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

- 898 • A virtual system snapshot is captured from a virtual system.
- 899 In this case, the virtual system snapshot becomes the most current snapshot in the branch. If no
- 900 branch exists, a new branch is created.

## 901 **7.7.5 Virtual system snapshot capabilities**

902 The provisions in this subclause are optional.

903 If virtual system snapshot capabilities are implemented, the provisions in this subclause apply.

904 This subclause models snapshot related capabilities of a virtual system in terms of the  
905 CIM\_VirtualSystemSnapshotCapabilities class.

### 906 **7.7.5.1 CIM\_VirtualSystemSnapshotCapabilities.SnapshotTypesEnabled[ ] array property**

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

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

### 912 **7.7.5.2 CIM\_VirtualSystemSnapshotCapabilities.GuestOSNotificationEnabled property**

913 The implementation of the GuestOSNotificationEnabled property is optional.

914 If the GuestOSNotificationEnabled property is implemented, the provisions in this subclause apply.

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

## 920 **8 Methods**

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

### 923 **8.1 General behavior of extrinsic methods**

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

#### 925 **8.1.1 Resource allocation requests**

926 Some methods specify the ResourceSettings[ ] array parameter. If set to a value other than NULL, each  
927 element of the ResourceSettings[ ] array parameter shall contain an embedded instance of the CIM\_Re-  
928 sourceAllocationSettingData class that describes a resource allocation request for a virtual resource or  
929 coherent set of virtual resources.

930 The use of the CIM\_ResourceAllocationSettingData class as input for operations is specified in [DSP1041](#).

931 One instance of the CIM\_ResourceAllocationSettingData class may affect one virtual resource or a coher-  
932 ent set of virtual resources. For example, one instance of CIM\_ResourceAllocationSettingData that has  
933 the value of the ResourceType property set to 3 (Processor) and the value of the VirtualQuantity property  
934 set to 2 requests the allocation of two virtual processors.

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

### 940 **8.1.2 Method results**

941 If a particular method is not implemented, a value of 1 (Not Supported) shall be returned.

942 If synchronous execution of a method succeeds, the implementation shall set a return value of  
943 0 (Completed with No Error).

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

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

- 948 • Set a return value of 4096 (Job Started).
- 949 • Set the value of the Job output parameter to refer to an instance of the CIM\_ConcreteJob class  
950 that represents the asynchronous task.
- 951 • Set the values of the JobState and TimeOfLastStateChange properties in that instance to repre-  
952 sent the state and last state change time of the asynchronous task.

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

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

- 956 • Set the value of the JobState property to 7 (Completed).
- 957 • Provide an instance of the CIM\_AffectedJobEntity association with property values set as fol-  
958 lows:
  - 959 – The value of the AffectedElement property shall refer to the object that represents the top-  
960 level entity that was created or modified by the asynchronous task. For example, for the  
961 DefineSystem( ) method, this is an instance of the CIM\_ComputerSystem class, and for  
962 the CreateSnapshot( ) method, this is an instance of the CIM\_VirtualSystemSettingData  
963 class that represents a snapshot of a virtual system.
  - 964 – The value of the AffectingElement property shall refer to the instance of the  
965 CIM\_ConcreteJob class that represents the completed asynchronous task.
  - 966 – The value of the first element in the ElementEffects[ ] array property (ElementEffects[0])  
967 shall be set to 5 (Create) for the DefineSystem( ) or CreateSnapshot( ) methods. Other-  
968 wise, this value shall be 0 (Unknown).

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

### 971 **8.1.3 Asynchronous processing**

972 An implementation may support asynchronous processing of some methods specified in the  
973 CIM\_VirtualSystemManagementService class.

974 **8.1.3.1 General requirements**

975 All of the following conditions shall be met:

- 976 • Elements that convey information about which methods of the  
977 CIM\_VirtualSystemManagementService class are implemented for asynchronous execution  
978 within an implementation are modeled in 7.4.4.
- 979 • Elements that convey information about which methods of the  
980 CIM\_VirtualSystemSnapshotService class are implemented for asynchronous execution within  
981 an implementation are modeled in 7.7.1.1.
- 982 • Elements that convey information about whether a method is executed asynchronously are  
983 modeled in 8.1.2.

984 **8.1.3.2 Job parameter**

985 The implementation shall set the value of the Job parameter as a result of an asynchronous execution of  
986 a method of the CIM\_VirtualSystemManagementService as follows:

- 987 • If the method execution is performed synchronously, the implementation shall set the value to  
988 NULL.
- 989 • If the method execution is performed asynchronously, the implementation shall set the value to  
990 refer to the instance of the CIM\_ConcreteJob class that represents the asynchronous task.

991 **8.2 Methods of the CIM\_VirtualSystemManagementService class**

992 This subclause models virtual system management services in terms of methods of the  
993 CIM\_VirtualSystemManagementService class.

994 **8.2.1 CIM\_VirtualSystemManagementService.DefineSystem( ) method**

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

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

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

999 The execution of the DefineSystem( ) method shall effect the creation of a new virtual system definition as  
1000 specified through the values of the SystemSettings parameter, the values of elements in the  
1001 ResourceSettings[ ] array parameter and elements of the configuration referred to by the value of the  
1002 ReferencedConfiguration parameter, and through default values that are established within the  
1003 implementation.

1004 Table 2 contains requirements for parameters of this method.

1005 **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.       |

### 1006 8.2.1.1 Value preference rules

1007 The DefineSystem( ) method facilitates the definition of a new virtual system at the host system, based on  
1008 client requirements specified through one or more virtual system configurations:

- 1009 • "Input" virtual system configuration

1010 The "Input" virtual system configuration is prepared locally by the client and provided in the form  
1011 of embedded instances of the CIM\_VirtualSystemSettingData class in the SystemSettings pa-  
1012 rameter and embedded instances of the CIM\_ResourceAllocationSettingData class as values  
1013 for elements of the ResourceSettings[ ] array parameter.

- 1014 • "Reference" virtual system configuration

1015 The "Reference" virtual system configuration is a "Defined" virtual system configuration that al-  
1016 ready exists within the implementation; it is referenced by the ReferencedConfiguration  
1017 parameter.

1018 An implementation shall define the virtual system based on "Input" and "Reference" configuration. It may  
1019 extend a virtual system definition beyond client requirements based on implementation-specific rules and  
1020 requirements.

1021 If only the "Reference" virtual system configuration is provided by the client, the implementation shall cre-  
1022 ate a copy or cloned configuration of the "Reference" virtual system configuration.

1023 If both configurations are provided by the client, the implementation shall give the "Input" virtual system  
1024 configuration preference over the "Reference" configuration. An implementation may support this behavior  
1025 at two levels:

- 1026 • The basic level supports the addition of resource allocations that were not requested by ele-  
1027 ments of the ResourceSettings[ ] array parameter, but that are defined in the "Reference" virtual  
1028 system configuration.
- 1029 • The advanced level, in addition, supports amending incomplete resource requests.

1030 In this case the correlation of instances of the CIM\_ResourceAllocationSettingData class in the  
1031 "Input" configuration and in the "Reference" configuration shall be established through the value  
1032 of the InstanceID parameter. If the value of the InstanceID parameter is identical for an instance  
1033 in the "Input" configuration and an instance in the "Reference" configuration, these instances to-  
1034 gether describe one virtual resource allocation request, such that non-NULL property values  
1035 specified in the "Input" configuration override those specified in the "Reference" configuration.

1036 If no value is specified for a property in the "Input" configuration or in the "Reference" configuration, the  
1037 implementation may exhibit an implementation-dependent default behavior. [DSP1059](#) and resource-type-  
1038 specific resource allocation DMTF management profiles may specify resource-type-specific behavior.

1039 If the DefineSystem( ) method is called without input parameters, the implementation may exploit a de-  
1040 fault behavior or may fail the method execution.

1041 NOTE: A client may inspect the "Reference" virtual system configuration before invoking the DefineSystem( )  
1042 method (see respective use cases in [DSP1057](#)).

### 1043 8.2.1.2 SystemSettings parameter

1044 A client should set the value of the SystemSettings parameter with an embedded instance of the  
1045 CIM\_VirtualSystemSettingData class that describes requested virtual system settings. The client may set  
1046 the value of the SystemSettings parameter to NULL, requesting the implementation to select input values  
1047 based on the rules specified in 8.2.1.1.

1048 An implementation shall interpret the value of the SystemSettings parameter as the system part of an  
1049 "Input" virtual system configuration, and apply the rules specified in 8.2.1.1.

1050 The use of the CIM\_VirtualSystemSettingData class as input for operations specified by this profile is  
 1051 specified in 10.22.

1052 **8.2.1.3 ResourceSettings[ ] array parameter**

1053 A client should set the ResourceSettings[ ] array parameter and apply the specifications given in 8.1.1.  
 1054 The client may set the value of the ResourceSettings[ ] array parameter to NULL or provide an empty ar-  
 1055 ray, requesting the implementation to define a default set of virtual resources (see 8.2.1.1).

1056 An implementation shall interpret the value of the ResourceSettings[ ] array parameter as the resource  
 1057 part of an "Input" virtual system configuration, and apply the value preference rules specified in 8.2.1.1.

1058 **8.2.1.4 ReferencedConfiguration parameter**

1059 A client may set a value of the ReferencedConfiguration parameter to refer to an existing "Defined" virtual  
 1060 system configuration. A client may set the value of the ReferencedConfiguration parameter to NULL, indi-  
 1061 cating that a "Reference" configuration shall not be used.

1062 An implementation shall use the "Reference" virtual system configuration according to the rules specified  
 1063 in 8.2.1.1.

1064 **8.2.1.5 ResultingSystem parameter**

1065 The implementation shall set the value of the ResultingSystem parameter as follows:

- 1066 • If the method execution is performed synchronously and is successful, the value is set to refer-  
 1067 ence the instance of the CIM\_ComputerSystem class that represents the newly defined virtual  
 1068 system.
- 1069 • If the method execution is performed synchronously and fails, or if the method execution is per-  
 1070 formed asynchronously, the value is set to NULL.

1071 **8.2.1.6 Return codes**

1072 An implementation shall indicate the result of the method execution by using the return code values speci-  
 1073 fied in Table 3.

1074 **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. |

1075 **8.2.2 CIM\_VirtualSystemManagementService.DestroySystem( ) method**

1076 The implementation of the DestroySystem( ) method is conditional.

1077 Condition: The definition and destruction of virtual systems is implemented; see 7.4.6.1.

1078 If the DestroySystem( ) method is implemented, the provisions in this subclause apply; in addition  
 1079 behavior applicable to all extrinsic methods is specified in 8.1.2.

1080 The execution of the DestroySystem( ) method shall effect the destruction of the referenced virtual system  
1081 and all related virtual system configurations, including snapshots.

1082 Table 4 contains requirements for parameters of this method.

1083 **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.       |

1084 **8.2.2.1 AffectedSystem parameter**

1085 A client shall set a value of the AffectedSystem parameter to refer to the instance of the  
1086 CIM\_ComputerSystem class that represents the virtual system to be destroyed.

1087 An implementation shall interpret the value of the AffectedSystem parameter to identify the virtual system  
1088 that is to be destroyed.

1089 **8.2.2.2 Return codes**

1090 An implementation shall indicate the result of the method execution by using the return code values speci-  
1091 fied in Table 5.

1092 **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.                             |

1093 **8.2.3 CIM\_VirtualSystemManagementService.AddResourceSettings( ) method**  
1094 **(Conditional)**

1095 The implementation of the AddResourceSettings( ) method is conditional.

1096 Condition: The addition and the removal of virtual resources to virtual systems is implemented; see  
1097 7.4.6.2.

1098 If the AddResourceSettings( ) method is implemented, the provisions in this subclause apply; in addition  
1099 behavior applicable to all extrinsic methods is specified in 8.1.2.

1100 The execution of the AddResourceSettings( ) method shall effect the entry of resource allocation requests  
1101 or resource allocations provided through the ResourceSettings[ ] array parameter in the affected virtual  
1102 system configuration.

1103 Table 6 contains requirements for parameters of this method.



1104

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

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

**1105 8.2.3.1 AffectedConfiguration parameter**

1106 A client shall set a value of AffectedConfiguration parameter to refer to the instance of the  
 1107 CIM\_VirtualSystemSettingData class that represents the virtual system configuration that receives new  
 1108 resource allocations.

1109 An implementation shall interpret the value of the AffectedConfiguration parameter to identify the virtual  
 1110 system configuration that receives new resource allocations.

**1111 8.2.3.2 ResourceSettings[ ] array parameter**

1112 A client shall set the ResourceSettings[ ] parameter containing one or more input instances of the  
 1113 CIM\_ResourceAllocationSettingData class as specified in a profile based on [DSP1041](#) and on [DSP1043](#),  
 1114 such as for example [DSP1044](#) or [DSP1047](#).

1115 If the value of the InstanceID property in any of the input CIM\_ResourceAllocationSettingData instances  
 1116 is other than NULL, that value shall be ignored; however, the remaining values of the input instance shall  
 1117 be respected as defined in the resource type specific resource allocation profile.

1118 An implementation shall apply the specifications given in 8.1.1.

**1119 8.2.3.3 ResultingResourceSettings[ ] array parameter**

1120 The implementation shall set the value of the ResultingResourceSettings[ ] array parameter as follows:

- 1121 • to an array of references to instances of the CIM\_ResourceAllocationSettingData class that  
 1122 represent resource allocations that were obtained during the execution of the method
- 1123 • to NULL, if the method is executed synchronously and fails, or if the method is executed  
 1124 asynchronously

**1125 8.2.3.4 Return codes**

1126 An implementation shall indicate the result of the method execution by using the return code values speci-  
 1127 fied in Table 7.

1128

**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. |

## 1129 8.2.4 CIM\_VirtualSystemManagementService.ModifyResourceSettings( ) method

1130 The implementation of the ModifyResourceSettings( ) method is conditional.

1131 Condition: The modification of virtual systems and resources is implemented; see 7.4.6.3.

1132 If the ModifyResourceSettings( ) method is implemented, the provisions in this subclause apply; in  
1133 addition behavior applicable to all extrinsic methods is specified in 8.1.2.

1134 If implemented, the execution of the ModifyResourceSettings( ) method shall effect the modification of re-  
1135 source allocation requests that exist, with the implementation using instances of the  
1136 CIM\_ResourceAllocationSettingData class that are passed in through values of elements of the  
1137 ResourceSettings[ ] array parameter.

1138 The execution of the ModifyResourceSettings( ) method shall effect the modification of resource alloca-  
1139 tions or resource allocation requests, such that non-key and non-NULL values of instances of the  
1140 CIM\_ResourceAllocationSettingData class provided as values for elements of the ResourceSettings[ ] ar-  
1141 ray parameter override respective values in instances identified through the InstanceID property.

1142 Table 8 contains requirements for parameters of this method.

1143 **Table 8 – ModifyResourceSettings( ) method: Parameters**

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

### 1144 8.2.4.1 ResourceSettings[ ] parameter

1145 The specifications in 8.1.1 apply.

1146 A client shall set the ResourceSettings[ ] parameter. Any instance of the  
1147 CIM\_ResourceAllocationSettingData class that is passed in as a value for elements of the  
1148 ResourceSettings[ ] array parameter shall conform to all of the following conditions:

1149 • It shall represent requests for the modification of virtual resource state extensions, virtual re-  
1150 source definitions scoped by one particular virtual system, or both.

1151 • It shall have a valid non-NULL value in the InstanceID property that identifies a respective in-  
1152 stance of the CIM\_ResourceAllocationSettingData class that represents an existing resource  
1153 allocation or resource allocation request within the implementation. This should be assured  
1154 through the execution of previously executed retrieve operations, such as the execution of  
1155 extrinsic methods or intrinsic CIM operations that yield respective instances of the  
1156 CIM\_ResourceAllocationSettingData class. For example, the client may use the intrinsic  
1157 GetInstance( ) CIM operation.

1158 The client shall modify such instances locally to reflect the desired modifications and finally pass  
1159 them back in as elements of the ResourceSettings[ ] array parameter. Modifications shall not be  
1160 applied to the InstanceID property that is the key property of the  
1161 CIM\_ResourceAllocationSettingData class. Further restriction may apply, such as from re-  
1162 source-type-specific resource allocation DMTF management profiles.

1163 An implementation shall apply the specifications given in 8.1.1. The implementation shall ignore any ele-  
1164 ment of the ResourceSettings[ ] array property that does not identify, through the value of the InstanceID  
1165 key property, an existing instance of the CIM\_ResourceAllocationSettingData class within the  
1166 implementation.

1167 **8.2.4.2 ResultingResourceSettings[ ] parameter**

1168 The implementation shall set the value of the ResultingResourceSettings[ ] array parameter as follows:

- 1169 • If the method was executed asynchronously, the value shall be set to NULL.
- 1170 • 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.
- 1171
- 1172
- 1173
- 1174 • If the method was executed synchronously and failed completely, the value shall be set to NULL.
- 1175

1176 **8.2.4.3 Return codes**

1177 An implementation shall indicate the result of the method execution by using the return code values specified in Table 9.

1179 **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. |   |

1180 **8.2.5 CIM\_VirtualSystemManagementService.ModifySystemSettings( ) method**

1181 The implementation of the ModifySystemSettings( ) method is conditional.

1182 Condition: The modification of virtual systems and resources is implemented; see 7.4.6.3.

1183 If the ModifySystemSettings( ) method is implemented, the provisions in this subclause apply; in addition behavior applicable to all extrinsic methods is specified in 8.1.2.

1185 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.

1189 Table 10 contains requirements for parameters of this method.

1190

**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.       |

**1191 8.2.5.1 SystemSettings parameter**

1192 A client shall set the SystemSettings parameter. Any instance of the CIM\_VirtualSystemSettingData class  
 1193 that is passed in as a value of the SystemSettings parameter shall have a valid non-NULL value in the  
 1194 InstanceID property that identifies a respective instance of the CIM\_VirtualSystemSettingData class  
 1195 existing within the implementation. A client shall obtain such an instance before invoking the  
 1196 ModifySystemSettings( ) method (for example, by using an extrinsic method or intrinsic CIM operation  
 1197 that yields a respective instance as a result). For example, the client may use the intrinsic GetInstance( )  
 1198 CIM operation. The client shall then modify the instance locally so that it reflects the desired modifications  
 1199 and finally pass it back in as a value of the SystemSettings parameter.

1200 The implementation shall ignore any value of the SystemSettings parameter that does not identify,  
 1201 through the value of the InstanceID key property, an existing instance of the  
 1202 CIM\_VirtualSystemSettingData class within the implementation.

**1203 8.2.5.2 Return codes**

1204 An implementation shall indicate the result of the method execution by using the return code values speci-  
 1205 fied in Table 11.

1206

**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.   |

**1207 8.2.6 CIM\_VirtualSystemManagementService.RemoveResourceSettings( ) method**

1208 The implementation of the RemoveResourceSettings( ) method is conditional.

1209 Condition: The addition and the removal of virtual resources to virtual systems is implemented; see  
 1210 7.4.6.2.

1211 If the RemoveResourceSettings( ) method is implemented, the provisions in this subclause apply; in  
 1212 addition behavior applicable to all extrinsic methods is specified in 8.1.2.

1213 The execution of the RemoveResourceSettings( ) method shall effect the removal of resource allocation  
 1214 requests identified by the value of elements of the ResourceSettings[ ] parameter.

1215 Table 12 contains requirements for parameters of this method.

1216 **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.       |

1217 **8.2.6.1 ResourceSettings[ ] array parameter**

1218 A client shall set the ResourceSettings[ ] array parameter. The value of any element specified in the  
 1219 ResourceSettings[ ] array parameter shall represent requests for the removal of virtual resource state  
 1220 extensions, of virtual resource definitions, or both in the scope of one virtual system.

1221 **8.2.6.2 Return codes**

1222 An implementation shall indicate the result of the method execution by using the return code values speci-  
 1223 fied in Table 13.

1224 **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. |

1225 **8.3 Methods of the CIM\_VirtualSystemSnapshotService class**

1226 This subclause models virtual system snapshot management in terms of methods of the  
 1227 CIM\_VirtualSystemSnapshotService class.

1228 **8.3.1 CIM\_VirtualSystemSnapshotService.CreateSnapshot( ) method**

1229 The implementation of the CreateSnapshot( ) method is conditional.

1230 Condition: The creation, destruction and application of virtual system snapshots is implemented; see  
 1231 7.7.1.1.

1232 If the CreateSnapshot( ) method is implemented, the provisions in this subclause apply; in addition  
 1233 behavior applicable to all extrinsic methods is specified in 8.1.2.

1234 The execution of the CreateSnapshot( ) method shall effect the creation of a snapshot of the affected vir-  
 1235 tual system. The snapshot shall have the type that is designated by the value of the SnapshotType  
 1236 parameter (see 8.3.1.3).

1237 A full snapshot shall contain all information required to restore the complete virtual system and its re-  
 1238 sources to exactly the situation that existed when the snapshot was created. Other types of snapshots  
 1239 may contain less information.

1240 If the virtual system is in the "Active" virtual system state, it may continue to perform tasks but may be  
1241 temporarily paused as the creation of the snapshot requires the capturing of state information.

1242 Table 14 contains requirements for parameters of this method.

1243 **Table 14 – CreateSnapshot( ) method: Parameters**

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

#### 1244 **8.3.1.1 AffectedSystem parameter**

1245 A client shall set a value of the AffectedSystem parameter to refer to the instance of the  
1246 CIM\_ComputerSystem class that represents the virtual system that is the source for the snapshot.

1247 An implementation shall interpret the value of the AffectedSystem parameter to identify the virtual system  
1248 that is the source for the snapshot.

#### 1249 **8.3.1.2 SnapshotSettings parameter**

1250 A client may set a value of the SnapshotSettings parameter with an embedded instance of a  
1251 CIM\_SettingData class. It is assumed that an implementation-specific class derived from  
1252 CIM\_SettingData contains additional implementation-specific properties that enable some control over  
1253 characteristics of the snapshot process.

1254 An implementation shall use the value of the SnapshotSettings parameter to control the characteristics of  
1255 the snapshot process.

#### 1256 **8.3.1.3 SnapshotType parameter**

1257 A client shall set the value of the SnapshotType parameter to designate the intended type of snapshot.  
1258 The value shall be one of the values set in the SnapshotTypesSupported[ ] array property in the instance  
1259 of the CIM\_VirtualSystemSnapshotServiceCapabilities class that is related to the snapshot service.

1260 An implementation shall use the value of the SnapshotType parameter to determine the requested type of  
1261 snapshot. If a value is not specified or is not one of the values set in the SnapshotTypesSupported[ ] array  
1262 property in the instance of the CIM\_VirtualSystemSnapshotServiceCapabilities class that is related to the  
1263 snapshot service, an implementation shall fail the method execution and set a return code of 6 (Invalid  
1264 Type).

#### 1265 **8.3.1.4 ResultingSnapshot parameter**

1266 The implementation shall set the value of the ResultingSnapshot parameter as follows:

- 1267 • If the method execution is performed synchronously and is successful, the value shall be set to  
1268 reference the instance of the CIM\_VirtualSystemSettingData class that represents the newly  
1269 created virtual system snapshot.
- 1270 • If the method execution is performed synchronously and fails, or if the method execution is per-  
1271 formed asynchronously, the value shall be set to NULL.

- 1272 • If the method execution is performed asynchronously and is successful, see 8.1.2 to locate the
- 1273 instance of the CIM\_VirtualSystemSettingData class that represents the newly created virtual
- 1274 system snapshot.

1275 **8.3.1.5 Return codes**

1276 An implementation shall indicate the result of the method execution by using the return code values speci-  
 1277 fied in Table 15.

1278 **Table 15 – CreateSnapshot( ) method: Return code values**

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

1279 **8.3.2 VirtualSystemSnapshotService.DestroySnapshot( ) method**

1280 The implementation of the DestroySnapshot( ) method is conditional.

1281 Condition: Virtual system snapshots are implemented; see 7.7.1.1.

1282 If the DestroySnapshot( ) method is implemented, the provisions in this subclause apply; in addition  
 1283 behavior applicable to all extrinsic methods is specified in 8.1.2.

1284 The execution of the DestroySnapshot( ) method shall effect the destruction of the affected virtual system  
 1285 snapshot. Dependency relationships from other snapshots to the affected snapshot shall be updated so  
 1286 that the affected snapshot is no longer referenced. If the snapshot was persistently established to be used  
 1287 during virtual system activation, the implementation may assign a different snapshot to be used for subse-  
 1288 quent virtual system activations, or may fall back to the "Default" virtual system configuration to be used  
 1289 for future activations. If a virtual system was activated using the snapshot and is still in a state other than  
 1290 the "Defined" virtual system state, the active virtual system shall not be affected by the execution of the  
 1291 DestroySnapshot( ) method.

1292 Table 16 contains requirements for parameters of this method.

1293 **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.       |

1294 **8.3.2.1 AffectedSnapshot parameter**

1295 A client shall set a value of the AffectedSnapshot parameter to refer to the instance of the  
1296 CIM\_VirtualSystemSettingData class that represents a snapshot.

1297 An implementation shall interpret the value of the AffectedSnapshot parameter to identify the snapshot  
1298 that is to be destroyed.

1299 **8.3.2.2 Return codes**

1300 An implementation shall indicate the result of the method execution using the return code values specified  
1301 by Table 17.

1302 **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.   |

1303 **8.3.3 VirtualSystemSnapshotService.ApplySnapshot( ) method**

1304 The implementation of the ApplySnapshot( ) method is conditional.

1305 Condition: Virtual system snapshots are implemented; see 7.7.1.1.

1306 If the ApplySnapshot( ) method is implemented, the provisions in this subclause apply; in addition  
1307 behavior applicable to all extrinsic methods is specified in 8.1.2.

1308 The execution of the ApplySnapshot( ) method shall indicate that the snapshot is used for the next  
1309 activation of the associated virtual system (the virtual system that was the source for the snapshot). The  
1310 method execution shall have one or both of the following effects:

- 1311 • The snapshot is persistently established to be used for subsequent activations.
- 1312 • The virtual system is immediately activated or recycled, using the snapshot.

1313 Table 18 contains requirements for parameters of this method.

1314 **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.       |



1315 **8.3.3.1 Snapshot parameter**

1316 A client shall set a value of the Snapshot parameter to refer to the instance of the  
 1317 CIM\_VirtualSystemSettingData class that represents a snapshot.

1318 An implementation shall interpret the value of the Snapshot parameter to identify the snapshot that is to  
 1319 be applied.

1320 **8.3.3.2 Return codes**

1321 An implementation shall indicate the result of the method execution by using the return code values speci-  
 1322 fied in Table 19.

1323 **Table 19 – ApplySnapshot( ) method: Return code values**

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

1324 **8.4 Profile conventions for operations**

1325 The default list of operations for all classes is:

1326     GetInstance( )

1327     EnumerateInstances( )

1328     EnumerateInstanceNames( )

1329 For classes that are referenced by an association, the default list also includes

1330     Associators( )

1331     AssociatorNames( )

1332     References( )

1333     ReferenceNames( )

1334 **8.4.1 CIM\_AffectedJobElement**

1335 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1336 NOTE Related profiles may define additional requirements on operations for the profile class.

**1337 8.4.2 CIM\_ComputerSystem**

1338 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1339 NOTE Related profiles may define additional requirements on operations for the profile class.

**1340 8.4.3 CIM\_ConcreteJob**

1341 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1342 NOTE Related profiles may define additional requirements on operations for the profile class.

**1343 8.4.4 CIM\_Dependency**

1344 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1345 NOTE Related profiles may define additional requirements on operations for the profile class.

**1346 8.4.5 CIM\_ElementCapabilities**

1347 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1348 NOTE Related profiles may define additional requirements on operations for the profile class.

**1349 8.4.6 CIM\_ElementConformsToProfile**

1350 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1351 NOTE Related profiles may define additional requirements on operations for the profile class.

**1352 8.4.7 CIM\_HostedDependency**

1353 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1354 NOTE Related profiles may define additional requirements on operations for the profile class.

**1355 8.4.8 CIM\_HostedService**

1356 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1357 NOTE Related profiles may define additional requirements on operations for the profile class.

**1358 8.4.9 CIM\_LastAppliedSnapshot**

1359 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1360 NOTE Related profiles may define additional requirements on operations for the profile class.

**1361 8.4.10 CIM\_MostCurrentSnapshotInBranch**

1362 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1363 NOTE Related profiles may define additional requirements on operations for the profile class.

**1364 8.4.11 CIM\_ReferencedProfile**

1365 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1366 NOTE Related profiles may define additional requirements on operations for the profile class.

**1367 8.4.12 CIM\_RegisteredProfile**

1368 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1369 NOTE Related profiles may define additional requirements on operations for the profile class.

**1370 8.4.13 CIM\_ServiceAffectsElement**

1371 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1372 NOTE Related profiles may define additional requirements on operations for the profile class.

**1373 8.4.14 CIM\_SnapshotOfVirtualSystem**

1374 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1375 NOTE Related profiles may define additional requirements on operations for the profile class.

**1376 8.4.15 CIM\_System**

1377 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1378 NOTE Related profiles may define additional requirements on operations for the profile class.

**1379 8.4.16 CIM\_VirtualSystemManagementCapabilities**

1380 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1381 NOTE Related profiles may define additional requirements on operations for the profile class.

**1382 8.4.17 CIM\_VirtualSystemManagementService**

1383 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1384 NOTE Related profiles may define additional requirements on operations for the profile class.

**1385 8.4.18 CIM\_VirtualSystemSnapshotService**

1386 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1387 NOTE Related profiles may define additional requirements on operations for the profile class.

**1388 8.4.19 CIM\_VirtualSystemSnapshotCapabilities**

1389 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1390 NOTE Related profiles may define additional requirements on operations for the profile class.

**1391 8.4.20 CIM\_VirtualSystemSnapshotServiceCapabilities**

1392 All operations in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1393 NOTE Related profiles may define additional requirements on operations for the profile class.

**1394 9 Use Cases**

1395 This clause contains informative text only.

1396 The following use cases and object diagrams illustrate use of this profile. They are for informational pur-  
1397 poses only and do not introduce behavioral requirements for implementations of the profile.

## 1398 **9.1 General assumptions**

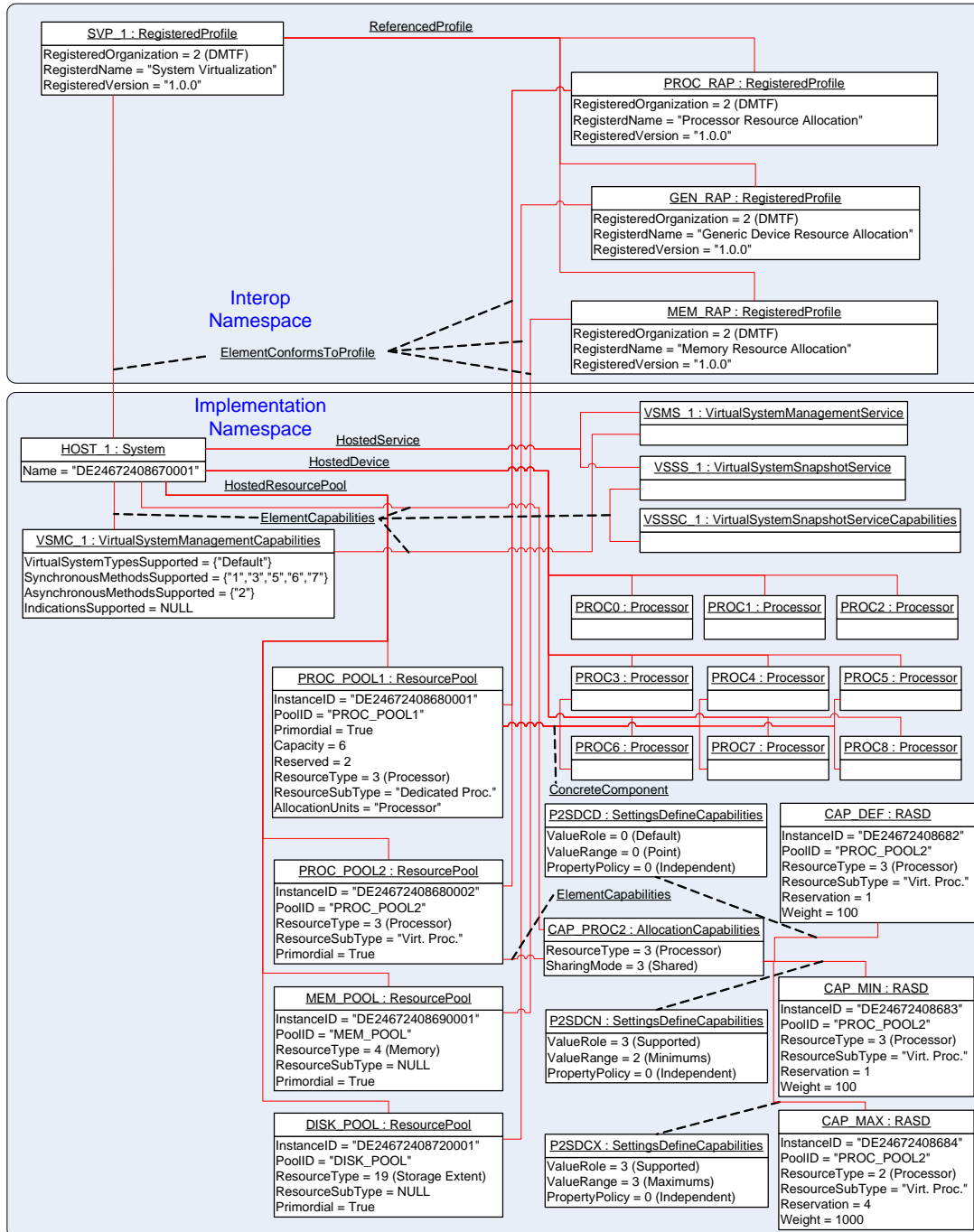
1399 For all use cases, it is assumed that a client performs intrinsic CIM operations, extrinsic CIM operations,  
1400 or both.

1401 For all use cases except the use case described in 9.2.1, the following conditions are implicitly assumed:

- 1402 • The client knows the URL of a WBEM service that exposes an implementation of this profile.
- 1403 • The client is able to communicate with the WBEM service through a specified CIM protocol. An  
1404 example is the use of the http protocol as described in [DSP0200](#). The client may use a facility  
1405 like a CIM client API to perform the encoding and decoding of CIM messages.

## 1406 **9.2 Discovery, localization, and inspection**

1407 This set of use cases describes how a client obtains access to an implementation, detects the central and  
1408 scoped instances, and analyzes information available through these instances. Figure 3 outlines a sample  
1409 situation that is referenced by some of the use-case descriptions in subsequent subclauses.



1410

1411 **Figure 3 – System Virtualization Profile instance diagram: Discovery, localization, and inspection**

1412 **9.2.1 SLP-Based discovery of CIM object managers hosting implementations of this**  
 1413 **Profile**

1414 The service location protocol (SLP) is used to locate WBEM services. A WBEM service that implements  
 1415 SLP as a discovery mechanism is required to register with SLP all instances of the CIM\_RegisteredProfile  
 1416 class that reside in the Interop namespace. An SLP service type is used to identify entities that are  
 1417 registered with SLP. An SLP service type is a structured string variable.

1418 **Assumption:** This profile is registered by at least one WBEM service that maintains a registration with an  
1419 SLP Directory Agent. The registration includes information about registered DMTF management profiles.  
1420 The client is able to make SLP calls.

- 1421 • The client invokes the SLPFindSrvs( ) SLP function as follows:
  - 1422 – The value of the srvtype parameter is set to "service:wbem".
  - 1423 – The value of the scopelist parameter is set to "default".
  - 1424 – The value of the filter parameter is set to "(RegisteredProfilesSupported=DMTF:System  
1425 Virtualization)".

1426 **Result:** Each URL in a list of URLs identifies a WBEM service where this profile is implemented.

### 1427 9.2.2 Locate conformant implementations using the EnumerateInstances( ) operation

1428 **Assumption:** The client knows the URL of a WBEM service hosting implementations of this profile (see  
1429 9.2.1).

- 1430 1) Using the URL, the client invokes the intrinsic EnumerateInstances( ) CIM operation with the  
1431 value of the ClassName input parameter set to "CIM\_RegisteredProfile".  
1432 The result is a list of instances of the CIM\_RegisteredProfile class.
- 1433 2) The client iterates over the list of instances of the CIM\_RegisteredProfile class and selects in-  
1434 stances where
  - 1435 – the RegisteredOrganization property has a value of 2 (DMTF)
  - 1436 – the RegisteredName property has a value of "System Virtualization"
  - 1437 – the RegisteredVersion property has a value equal to or greater than "1.0.0"

1438 **Result:** The client knows a set of instances of the CIM\_RegisteredProfile class, each representing an im-  
1439 plementation of this profile.

1440 In the example shown in Figure 3, one instance of the CIM\_RegisteredProfile class represents an imple-  
1441 mentation of this profile; it is tagged SVP\_1.

### 1442 9.2.3 Locate conformant implementations using the ExecuteQuery( ) operation

1443 **Assumption:** The client knows the URL of a WBEM service hosting implementations of this profile (see  
1444 9.2.1).

- 1445 • Using the URL, the client invokes the intrinsic ExecuteQuery( ) CIM operation as follows:
  - 1446 – The value of the QueryLanguage input parameter is set to "CIM:CQL".
  - 1447 – The value of the Query input parameter is set to "SELECT \* FROM CIM\_RegisteredProfile  
1448 WHERE RegisteredName = 'System Virtualization' AND RegisteredVersion >= '1.0.0'".

1449 **Result:** The client knows a set of instances of the CIM\_RegisteredProfile class, each representing an im-  
1450 plementation of this profile.

1451 In the example shown in Figure 3, one instance of the CIM\_RegisteredProfile class represents an imple-  
1452 mentation of this profile; it is tagged SVP\_1.

### 1453 9.2.4 Locate host systems represented by central instances of this profile

1454 **Assumption:** The client knows a reference to an instance of the CIM\_RegisteredProfile class that  
1455 represents an implementation of this profile (see 9.2.2 or 9.2.3).

- 1456 • The client invokes the intrinsic AssociatorNames( ) CIM operation as follows:

- 1457           – The value of the ObjectName parameter is set to refer to the instance of the  
1458           CIM\_RegisteredProfile class.
- 1459           – The value of the AssocClass parameter is set to "CIM\_ElementConformsToProfile".
- 1460           – The value of the ResultClass parameter is set to "CIM\_System".

1461 **Result:** The client knows a set of references to instances of the CIM\_System class that represent host  
1462 systems that are central and scoping instances of this profile.

1463 In the example shown in Figure 3, one instance of the CIM\_RegisteredProfile class represents a host sys-  
1464 tem that is a central and scoping instance of this profile; it is tagged HOST\_1.

### 1465 9.2.5 Locate implementations of scoped resource allocation profiles

1466 **Assumption:** The client knows a reference to an instance of the CIM\_RegisteredProfile class that  
1467 represents an implementation of this profile (see 9.2.2 or 9.2.3).

1468           1) The client invokes the intrinsic Associators( ) CIM operation to obtain a the list of scoped DMTF  
1469 management profiles, as follows:

- 1470           – The value of the ObjectName parameter is set to refer to the instance of the  
1471           CIM\_RegisteredProfile class.
- 1472           – The value of the AssocClass parameter is set to "CIM\_ReferencedProfile".
- 1473           – The value of the ResultClass parameter is set to "CIM\_RegisteredProfile".

1474           The result is a set of instances of the CIM\_RegisteredProfile class that each represent an imple-  
1475           mentation of a DMTF management profile that is scoped by this profile.

1476           2) For each instance of the CIM\_RegisteredProfile class, the client determines whether the value  
1477           of the RegisteredName property matches the registered name of one of the scoped resource  
1478           allocation DMTF management profiles as specified by Table 1.

1479           If the value does not match any name of a resource allocation DMTF management profile  
1480           scoped by this profile, the client ignores that instance of the CIM\_RegisteredProfile class.

1481 **Result:** The client knows a set of instances of the CIM\_RegisteredProfile class that each represent an im-  
1482           plementation of a resource allocation DMTF management profile that is scoped by this profile.

1483 In the example shown in Figure 3, three instances of the CIM\_RegisteredProfile class are associated with  
1484 the instance of the CIM\_RegisteredProfile class that is tagged SVP\_1 and represents a central instance  
1485 of this profile. These instances represent implementations of scoped resource allocation DMTF  
1486 management profiles:

- 1487           • The instance tagged PROC\_RAP represents an implementation of [DSP1044](#).
- 1488           • The instance tagged GEN\_RAP represents an implementation of [DSP1059](#).
- 1489           • The instance tagged MEM\_RAP represents an implementation of [DSP1045](#).

### 1490 9.2.6 Locate virtual system management service

1491 **Assumption:** The client knows a reference to an instance of the CIM\_System class that represents a  
1492 host system that is a central instance of this profile (see 9.2.4).

1493           • The client invokes the intrinsic AssociatorNames( ) CIM operation as follows:

- 1494           – The value of the ObjectName parameter is set to refer to the instance of the CIM\_System  
1495           class.
- 1496           – The value of the AssocClass parameter is set to "CIM\_HostedService".
- 1497           – The value of the ResultClass parameter is set to "CIM\_VirtualSystemManagementService".

1498 **Result:** The client knows a reference to the instance of the CIM\_VirtualSystemManagementService class  
 1499 that represents the virtual system management service that serves the host system. If the operation is  
 1500 successful, the size of the result set is 1.

1501 In the example shown in Figure 3, one instance of the CIM\_VirtualSystemManagementService class  
 1502 serves the host system; it is tagged VSMS\_1.

### 1503 9.2.7 Determine the capabilities of an implementation

1504 **Assumption:** The client knows a reference to an instance of the CIM\_System class that represents a  
 1505 host system that is a central instance of this profile (see 9.2.4).

- 1506 1) The client invokes the intrinsic Associators( ) CIM operation as follows:
- 1507 – The value of the ObjectName parameter is set to refer to the instance of the CIM\_System
  - 1508 class.
  - 1509 – The value of the AssocClass parameter is set to "CIM\_ElementCapabilities".
  - 1510 – The value of the ResultClass parameter is set to
  - 1511 "CIM\_VirtualSystemManagementCapabilities".

1512 The result is a list of instances of the CIM\_VirtualSystemManagementCapabilities class. If the  
 1513 operation is successful, the size of the result set is 1.

- 1514 2) The client analyzes the instance of the CIM\_VirtualSystemManagementCapabilities class.
- 1515 – The VirtualSystemTypesSupported[ ] array property lists identifiers of virtual system types
  - 1516 that the implementation supports.
  - 1517 – The SynchronousMethodsSupported[ ] array property lists identifiers of methods of the
  - 1518 CIM\_VirtualSystemManagementService class that are implemented with synchronous
  - 1519 method execution only.
  - 1520 – The AsynchronousMethodsSupported[ ] array property lists identifiers of methods of the
  - 1521 CIM\_VirtualSystemManagementService class that are implemented with synchronous and
  - 1522 asynchronous method execution.
  - 1523 – The IndicationsSupported[ ] array property lists identifiers of types of indications that the
  - 1524 implementation supports.

1525 **Result:** The client knows the capabilities of the host system in terms of properties of the  
 1526 CIM\_VirtualSystemManagementCapabilities class.

1527 In the example shown in Figure 3, one instance of the CIM\_VirtualSystemManagementCapabilities class  
 1528 is associated with the host system; it is tagged VSMC\_1.

- 1529 • The VirtualSystemTypesSupported[ ] array property lists one element with the value "Default",  
 1530 which indicates that the implementation supports one virtual system type named "Default". The  
 1531 semantics are implementation specific.
- 1532 • The SynchronousMethodsSupported[ ] array property lists enumerated values:  
 1533 { 1 (AddResourceSettingsSupported), 3 (DestroySystemSupported),  
 1534 5 (ModifyResourceSettingsSupported), 6 (ModifySystemSettingsSupported), and  
 1535 7 (RemoveResourcesSupported) }, which indicates that the AddResources( ) method, the  
 1536 DestroySystem( ) method, the ModifyResourceSettings( ) method, and the  
 1537 RemoveResourceSettings( ) method are implemented by the implementation with synchronous  
 1538 execution.
- 1539 • The AsynchronousMethodsSupported[ ] array property lists the enumerated value  
 1540 { 2 (DefineSystemSupported) }, which indicates that the DefineSystem( ) method is  
 1541 implemented by the implementation with synchronous or asynchronous execution.



- 1542 • The value of the IndicationsSupported[ ] array property is NULL, which indicates that indications  
1543 are not implemented by the implementation.

### 1544 9.2.8 Locate hosted resource pools of a particular resource type

1545 **Assumption:** The client knows a reference to an instance of the CIM\_System class that represents a  
1546 host system that is a central instance of this profile (see 9.2.4).

- 1547 1) The client invokes the intrinsic Associators( ) CIM operation as follows:
  - 1548 – The value of the ObjectName parameter is set to refer to the instance of the CIM\_System  
1549 class.
  - 1550 – The value of the AssocClass parameter is set to "CIM\_HostedResourcePool".
  - 1551 – The value of the ResultClass parameter is set to "CIM\_ResourcePool".

1552 The result is a list of instances of the CIM\_ResourcePool class.
- 1553 2) For each instance of CIM\_ResourcePool, the client determines whether the value of the  
1554 ResourceType property matches the requested resource type.
  - 1555 If the value does not match the requested resource type, the client drops that instance of the  
1556 CIM\_ResourcePool class from the list.

1557 **Result:** The client knows a set of instances of the CIM\_ResourcePool class, each representing a hosted  
1558 resource pool of the requested resource type.

### 1559 9.2.9 Obtain a set of central instances of scoped resource allocation profiles

1560 Resource allocation DMTF management profiles are based on [DSP1041](#) that defines the  
1561 CIM\_ResourcePool class as the central class. The procedure for the determination of central instances of  
1562 scoped DMTF management profiles depends on the profile advertisement methodology applied by the  
1563 respective implementations.

1564 **Assumption:** The client knows a reference to an instance of the CIM\_RegisteredProfile class that  
1565 represents an implementation of a scoped DMTF management profile (see 9.2.5).

- 1566 • The client invokes the intrinsic Associators( ) CIM operation to obtain the list of instances of the  
1567 CIM\_ResourcePool class that are central instances of the scoped DMTF management profiles,  
1568 as follows:
  - 1569 – The value of the ObjectName parameter is set to refer to the instance of the  
1570 CIM\_RegisteredProfile class
  - 1571 – The value of the AssocClass parameter is set to "CIM\_ElementConformsToProfile".
  - 1572 – The value of the ResultClass parameter is set to "CIM\_ResourcePool".

1573 The result is a list of instances of the CIM\_ResourcePool class; the list may be empty.

  - 1574 – If the list is not empty, the *central class* profile implementation advertisement methodology  
1575 is applied by the implementation for the scoped resource allocation DMTF management  
1576 profile. In this case, the list is the result for this use case.
  - 1577 – If the list is empty, the *scoping class* profile implementation advertisement methodology is  
1578 applied by the implementation for the scoped resource allocation DMTF management pro-  
1579 file. In this case, the client
    - 1580 – needs to know the resource type associated with the scoped resource allocation  
1581 DMTF management profile

- 1582                   – applies use case 9.2.8 to obtain a list of instances of the CIM\_ResourcePool class  
1583                   that each represent a resource pool of that particular resource type.

1584                   The resulting list is the result for this use case.

1585 **Result:** The client knows a list of instances of the CIM\_ResourcePool class, each representing a central  
1586 instance of a scoped resource allocation DMTF management profile.

### 1587 **9.2.10 Determine implemented resource types**

1588 **Assumption:** The client knows a reference to an instance of the CIM\_RegisteredProfile class that  
1589 represents an implementation of this profile (see 9.2.2 or 9.2.3).

- 1590           1) The client locates implementations of DMTF management profiles that are scoped by this profile  
1591           (see 9.2.5).

1592           The result is a list of references to instances of the CIM\_RegisteredProfile class that represent  
1593           implementations of DMTF management profiles that are scoped by this profile.

- 1594           2) For each instance of CIM\_RegisteredProfile, the client obtains the set of instances of the  
1595           CIM\_ResourcePool class that are central instances of the respective scoped resource allocation  
1596           DMTF management profiles and represent a conformant resource pool (see 9.2.9).

1597           The result is a list of instances of the CIM\_ResourcePool class that are central instances of  
1598           scoped resource allocation DMTF management profiles.

- 1599           3) The client creates an initially empty list of integer values. For each instance that is a result from  
1600           step 2), the client determines whether the value of property ResourceType is already repre-  
1601           sented in the list:

1602           – If that value is already contained in the list, the client ignores the element.

1603           – If that value is not yet contained in the list, the client adds a new element to the list with  
1604           that value.

1605 **Result:** The client knows a list of integer values, each designating a resource type that is supported by  
1606 the implementation.

1607 In the example shown in Figure 3, three instances of the CIM\_RegisteredProfile class are associated with  
1608 the instance of the CIM\_RegisteredProfile class that represents the implementation of this profile. These  
1609 instances are central instances of scoped resource allocation DMTF management profiles:

- 1610           • The instance tagged PROC\_RAP represents an implementation of [DSP1044](#).
- 1611           • The instance tagged GEN\_RAP represents an implementation of [DSP1059](#).
- 1612           • The instance tagged MEM\_RAP represents an implementation of [DSP1045](#).

1613 These instances are all associated with respective instances of the CIM\_ResourcePool class, indicating  
1614 that in this example in all cases the central class profile advertisement methodology is in use:

- 1615           • The instance tagged PROC\_RAP is associated with two instances that represent resource  
1616           pools for the allocation of processors. They show a value of 3 (Processor) for the ResourceType  
1617           property and are tagged PROC\_POOL1 and PROC\_POOL2.
- 1618           • The instance tagged GEN\_RAP is associated with one instance that represents a resource pool  
1619           for the allocation of virtual disks. It shows a value of 19 (Storage Extent) for the ResourceType  
1620           property and is tagged DISK\_POOL.
- 1621           • The instance tagged MEM\_RAP is associated with one instance that represents a resource pool  
1622           for the allocation of memory. It shows a value of 4 (Memory) for the ResourceType property and  
1623           is tagged MEM\_POOL.

1624 The resulting list of integer values is {"3","4","19"} and designates the implemented resource types  
 1625 3 (Processor), 4 (Memory), and 19 (Storage Extent).

### 1626 9.2.11 Determine the default resource pool for a resource type

1627 **Assumption:** The client knows a reference to an instance of the CIM\_System class that represents a  
 1628 host system that is a central instance of this profile (see 9.2.4).

1629 1) The client invokes the intrinsic Associators( ) CIM operation for a list of allocation capabilities  
 1630 associated with resource pools hosted by the host system, as follows:

- 1631 – The value of the ObjectName parameter is set to refer to the instance of the CIM\_System  
 1632 class.
- 1633 – The value of the AssocClass parameter is set to "CIM\_ElementCapabilities".
- 1634 – The value of the ResultClass parameter is set to "CIM\_AllocationCapabilities".

1635 The result is a list of instances of the CIM\_AllocationCapabilities class.

1636 2) The client drops instances from the result list of step 1) that have a value for the ResourceType  
 1637 property that does not match the requested resource type.

1638 The purpose of the following two steps is to further limit the result set from step 2) to those in-  
 1639 stances of the CIM\_AllocationCapabilities class that describe default settings. Default settings  
 1640 are flagged in the connecting instance of the CIM\_ElementCapabilities association that has a  
 1641 value of 2 (Default) for the Characteristics property.

1642 3) For each instance of the list resulting from step 2), the client invokes the intrinsic References( )  
 1643 CIM operation for a list of association instances that refer to the resource pool:

- 1644 – The value of the ObjectName parameter refers the instance of the CIM\_ResourcePool  
 1645 class.
- 1646 – The value of the ResultClass parameter is set to "CIM\_AllocationCapabilities".

1647 The result is a list of instances of the CIM\_ElementCapabilities association that associate an in-  
 1648 stance of the CIM\_ResourcePool class that is taken from the result of step 2).

1649 4) From the list obtained in step 3), the client drops all elements that meet either of the following  
 1650 conditions:

- 1651 – have a value other than 2 (Default) for the Characteristics property
- 1652 – do not refer to the instance of the CIM\_System class that represents the host system  
 1653 through the ManagedElement property

1654 The list should now contain one instance of the CIM\_AllocationCapabilities class that represents  
 1655 default allocation capabilities for the resource type in question.

1656 5) The client invokes the intrinsic Associators( ) CIM operation to resolve association for the re-  
 1657 source pool, as follows:

- 1658 – The value of the ObjectName parameter refers to the instance of the  
 1659 CIM\_AllocationCapabilities class selected in step 4).
- 1660 – The value of the AssocClass parameter is set to "CIM\_ElementCapabilities".
- 1661 – The value of the ResultClass parameter is set to "CIM\_ResourcePool".

1662 The result is a list of instances of the CIM\_ResourcePool class. The size of the list is 1.

1663 **Result:** The client knows the instance of the CIM\_ResourcePool class that represents the default re-  
 1664 source pool for the requested resource type.

1665 In the example shown in Figure 3, allocation capabilities are depicted only for the virtual processor pool.  
1666 In the subsequent description, it is assumed that the client looks for the default resource pool for proces-  
1667 sors:

- 1668 • With step 1) of this use case, the client resolves the CIM\_ElementCapabilities association from  
1669 the instance of the CIM\_System class that represents the host system (tagged HOST\_1) to in-  
1670 stances of the CIM\_AllocationCapabilities class. A conformant implementation of [DSP1043](#)  
1671 shows only one associated element for each resource type.
- 1672 • With step 2), the client reduces the result set to the one element that describes allocation  
1673 capabilities processors. This instance is tagged CAP\_PROC1.
- 1674 • With steps 3) and 4), the client further reduces the result set to the one instance of the  
1675 CIM\_AllocationCapabilities class that represents the system's default capabilities for resource  
1676 type 3 (Processor).
- 1677 • With step 5), the client resolves the CIM\_ElementCapabilities association in order to obtain the  
1678 instance of the CIM\_ResourcePool class that represents the default resource pool for  
1679 processors. This instance is tagged PROC\_POOL2.

### 1680 **9.2.12 Determine the resource pool for a resource allocation request or an allocated** 1681 **resource**

1682 **Assumption:** The client knows a reference to an instance of the CIM\_ResourceAllocationSettingData  
1683 class that represents a resource allocation request or allocated resource.

- 1684 • The client invokes the intrinsic Associators( ) CIM operation for a list of allocation capabilities  
1685 associated with resource pools hosted by the host system, as follows:
  - 1686 – The value of the ObjectName parameter is set to refer to the instance of the  
1687 CIM\_ResourceAllocationSettingData class.
  - 1688 – The value of the AssocClass parameter is set to "CIM\_ResourceAllocationFromPool".
  - 1689 – The value of the ResultClass parameter is set to "CIM\_ResourcePool".

1690 The result is a list of instances of the CIM\_ResourcePool class containing one element.

1691 **Result:** The client knows the instance of the CIM\_ResourcePool class that represents the resource pool  
1692 for the resource allocation request or allocated resource.

### 1693 **9.2.13 Determine valid settings for a resource type**

1694 This use case describes the determination of valid settings for a resource type in the context of either the  
1695 system as a whole or one resource pool.

1696 **Assumption:** The client knows a reference to either of the following instances:

- 1697 • an instance of the CIM\_ResourcePool class that represents a resource pool that is a central in-  
1698 stance of a resource allocation DMTF management profile
- 1699 • an instance of the CIM\_System class that represents a host system

1700 The sequence of activities is as follows:

- 1701 1) The client invokes the intrinsic Associators( ) CIM operation as follows:
  - 1702 – The value of the ObjectName parameter is set to refer to the instance of the  
1703 CIM\_ResourcePool class or the CIM\_System class.
  - 1704 – The value of the AssocClass parameter is set to "CIM\_ElementCapabilities".
  - 1705 – The value of the ResultClass parameter is set to "CIM\_AllocationCapabilities".

- 1706 The result is a list of instances of the CIM\_AllocationCapabilities class that describe the  
1707 capabilities of the input instance.
- 1708 2) The client drops from the result of step 1) those instances in which the ResourceType property  
1709 designates a resource type other than the requested resource type. This step is required only if  
1710 the starting point of the use case was an instance of the CIM\_System class.
- 1711 At this point the client has a list of instances of the CIM\_AllocationCapabilities class that de-  
1712 scribe allocation capabilities. The value of the SharingMode property allows a distinction  
1713 between shared and dedicated resources.
- 1714 3) The client invokes the intrinsic References( ) CIM operation for a set of instances of the  
1715 CIM\_SettingsDefineCapabilities association that each associate one instance of the  
1716 CIM\_ResourceAllocationSettingData class that describes a limiting aspect (min/max/increment),  
1717 as follows:
- 1718 – The value of the ObjectName parameter is set to refer to the instance of the  
1719 CIM\_AllocationCapabilities class.
  - 1720 – The value of the ResultClass parameter is set to "CIM\_SettingsDefineCapabilities".
- 1721 The result is a list of instances of the CIM\_SettingsDefineCapabilities association.
- 1722 4) For each instance that is a result from step 3), the client analyzes the values of the  
1723 PropertyPolicy property and the ValueRange property. The value of the ValueRole property is  
1724 irrelevant in this case.
- 1725 The property values have the following impact:
- 1726 – The value of the PropertyPolicy property is 0 (Independent) for a conformant  
1727 implementation of [DSP1043](#) in association instances that connect a min/max/increment  
1728 limiting setting.
  - 1729 – The value of the ValueRange property allows determining the designation of the associated  
1730 setting:
    - 1731 – A value of 1 (Minimums) indicates that the referenced instance of the  
1732 CIM\_ResourceAllocationSettingData class represents a lower limit for the allocation of  
1733 resources of the respective resource type.
    - 1734 – A value of 2 (Maximums) indicates that the referenced instance of the  
1735 CIM\_ResourceAllocationSettingData class represents an upper limit for the allocation  
1736 of resources of the respective resource type.
    - 1737 – A value of 3 (Increments) indicates that the referenced instance of the  
1738 CIM\_ResourceAllocationSettingData class represents an increment for the allocation  
1739 of resources of the respective resource type.
- 1740 5) For each association instance obtained in step 4), the client invokes the intrinsic GetInstance( )  
1741 CIM operation for the instance of the CIM\_ResourceAllocationSettingData class that describes  
1742 the respective limitation. The value of InstanceName parameter is set to the value of the  
1743 PartComponent property in the association instance obtained in step 4).
- 1744 In each case, the result is an instance of the CIM\_ResourceAllocationSettingData class that  
1745 represents a limiting setting.

1746 **Result:** The client knows the valid resource settings for the requested resource type.

#### 1747 9.2.14 Determine implementation class specifics

1748 This profile specifies the use of classes derived from the CIM\_SettingData class, namely the  
1749 CIM\_VirtualSystemSettingData class and the CIM\_ResourceAllocationSettingData class. Instances of  
1750 these classes are used to describe requirements on virtual systems and virtual resources as these are  
1751 created or modified. An implementation may provide platform-specific implementation classes that extend

1752 these classes (or, for the CIM\_ResourceAllocationSettingData class, that extend resource-type-specific  
1753 extensions specified in a resource-type-specific resource allocation DMTF management profile).

1754 A client should be prepared to deal with these extensions. A client should obtain class information for all  
1755 derived classes it deals with, in particular focusing on all class qualifiers and all property qualifiers,  
1756 namely

- 1757 • the Description qualifier that provides a description of the subclass or property
- 1758 • the DisplayName qualifier that provides a name for each subclass or property that is potentially  
1759 known to end-users

1760 **Assumption:** The client knows a reference to an instance of the class for which the client wants to obtain  
1761 class-specific information.

- 1762 1) The client extracts the class name from the reference.
  - 1763 2) The client invokes the intrinsic GetClass( ) CIM operation to obtain a formal class description,  
1764 as follows:
    - 1765 – The value of the ClassName parameter is set to the name of the class.
    - 1766 – The value of the LocalOnly parameter is set to "false".
    - 1767 – The value of the IncludeQualifiers parameter is set to "true".
    - 1768 – The value of the IncludeClassOrigin parameter is set to "true".
- 1769 The result is a description of a CIM class.

1770 **Result:** The client has a description of the class. The format depends on the CIM client used to issue the  
1771 request and is based on the XML class data structure that describes a CIM class as defined in [DSP0201](#).  
1772 The description contains the class's qualifiers, its properties with property qualifiers, and its methods with  
1773 method qualifiers. Inspection of the class description enables the client to create local instances of the  
1774 respective implementation class.

### 1775 9.2.15 Determine the implementation class for a resource type

1776 **Assumption:** The client knows a list of references to instances of the CIM\_ResourcePool class that  
1777 represent resource pools available at a host system.

- 1778 1) The client applies use case 9.2.13 to obtain a reference to an instance of the  
1779 CIM\_ResourceAllocationSettingData class that is associated with an instance of the  
1780 CIM\_ResourcePool class of the requested type through an instance of the  
1781 CIM\_SettingsDefineCapabilities association with the ValueRole property set to "DEFAULT".
- 1782 2) The client applies use case 9.2.14 to obtain class information about that instance.

1783 **Result:** The client has an implementation class descriptor, which allows the client to analyze the  
1784 implementation class for its qualifiers, its properties and their qualifiers, and its methods and their  
1785 qualifiers. Further, the client can create local instances of the returned class that may be used as input on  
1786 methods of the CIM\_VirtualSystemManagementService class.

### 1787 9.2.16 Locate virtual systems hosted by a host system

1788 **Assumption:** The client knows a reference to an instance of the CIM\_System class that is the central in-  
1789 stance of this profile and represents a host system (see 9.2.4).

- 1790 • The client invokes the intrinsic AssociatorNames( ) CIM operation for the list of virtual systems,  
1791 as follows:
  - 1792 – The value of the ObjectName parameter is set to refer to the instance of the CIM\_System  
1793 class.

- 1794 – The value of the AssocClass parameter is set to "CIM\_HostedSystem".
- 1795 – The value of the ResultClass parameter is set to "CIM\_ComputerSystem".
- 1796 The result is a list of references to instances of the CIM\_ComputerSystem class.

1797 **Result:** The client knows a set of references to instances of the CIM\_ComputerSystem class that  
 1798 represent virtual systems that are hosted by the host system.

### 1799 9.3 Virtual system definition, modification, and destruction

1800 **General assumption:** The client knows a reference to an instance of the  
 1801 CIM\_VirtualSystemManagementService class that represents the virtual system management services of  
 1802 a host system (see 9.2.6).

#### 1803 9.3.1 Virtual system definition

1804 Virtual system definition is performed using a client-provided configuration, a configuration of an existing  
 1805 virtual system, a configuration that is stored within the implementation, or combinations of these.

##### 1806 9.3.1.1 Define virtual system based on input and reference virtual system configuration

1807 **Assumption:** No assumption is made beyond the general assumption specified in 9.3.

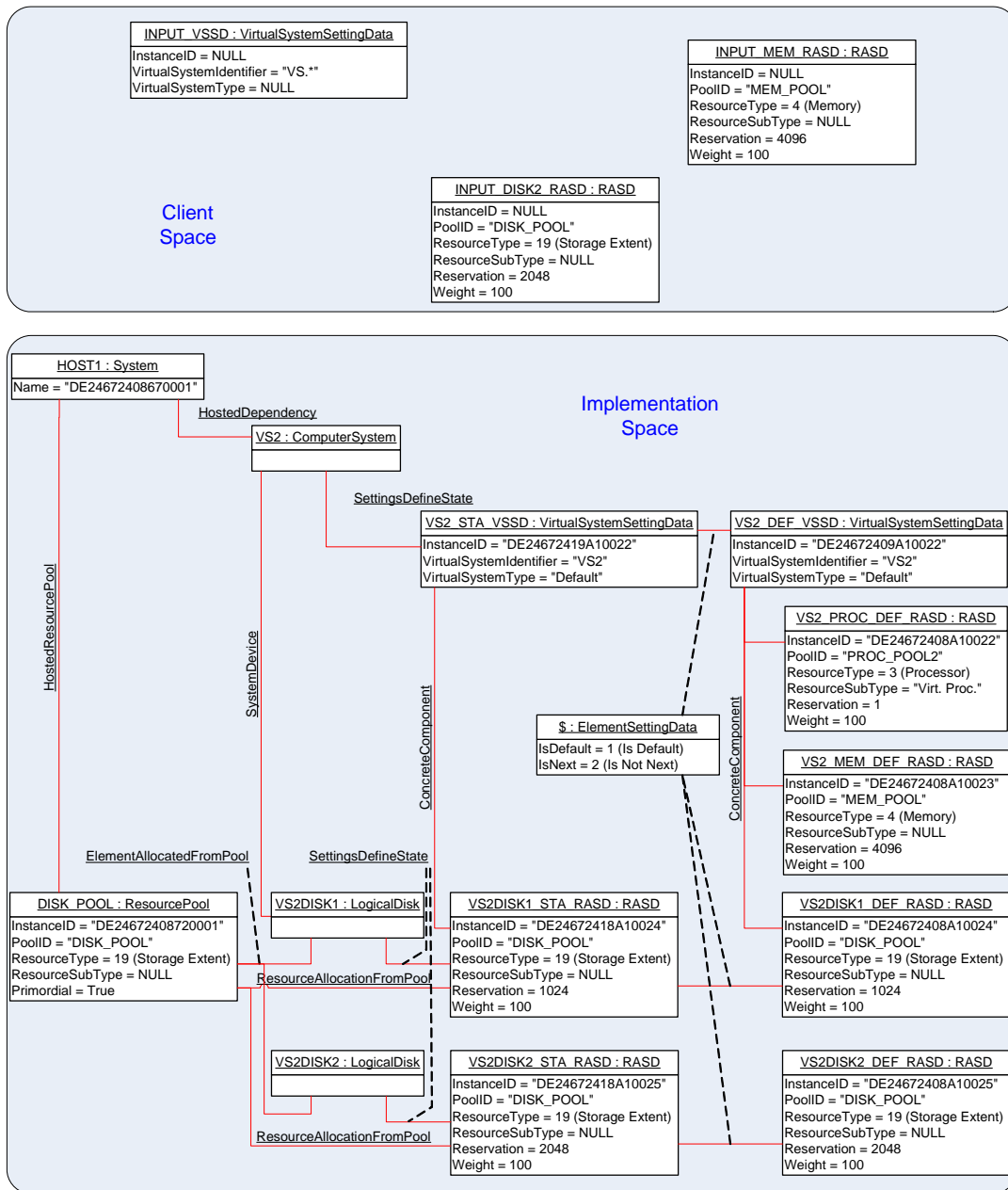
- 1808 1) The client invokes the DefineSystem( ) method (see 8.2.1) on the virtual system management  
 1809 service, as follows.
  - 1810 – The value of the SystemSettings parameter is set to an embedded instance of the  
 1811 CIM\_VirtualSystemSettingData class.
  - 1812 – The value of the ResourceSettings[ ] array parameter is set to an array of embedded in-  
 1813 stances of the CIM\_ResourceAllocationSettingData class.
  - 1814 – The value of the ReferenceConfiguration parameter is set to refer to a "Reference" virtual  
 1815 system configuration.
- 1816 2) The implementation executes the DefineSystem( ) method. The configuration of the new virtual  
 1817 system is created according to the client's requirements. The new virtual system is in the  
 1818 "Defined" virtual system state.
 

1819 The value returned in the ResultingSystem parameter refers to an instance of the  
 1820 CIM\_ComputerSystem class.

1821 **Result:** The client knows a reference to an instance of the CIM\_ComputerSystem class that represents  
 1822 the new virtual system.

1823 Figure 4 shows the representation of a virtual system that was defined using an "Input" virtual system and  
 1824 a "Reference" virtual system configuration.





1825

1826  
1827

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

1828  
1829  
1830  
1831  
1832

The new virtual system is represented by an instance of the CIM\_ComputerSystem class that is tagged VS2. The right side of Figure 4 shows the "Defined" virtual system configuration for the new virtual system. It is based on the "Input" virtual system configuration shown at the top of Figure 4. In this example, it is assumed that the ReferenceConfiguration parameter refers to a virtual system configuration that contains requests for the following resources:

1833  
1834  
1835

- a virtual processor
- virtual memory of 1024 MB
- a virtual disk of 1024 MB



1836 The "Input" virtual system configuration does not request the allocation of a processor, but because the  
 1837 "Reference" virtual configuration does, the resulting virtual system definition contains a request for a  
 1838 processor as well.

1839 The input virtual system configuration requests 4096 MB of memory. That value is given preference over  
 1840 the value of 1024 that is specified in the "Reference" configuration.

1841 The input virtual system configuration requests a virtual disk in addition to the one requested by the  
 1842 "Reference" configuration, resulting in two virtual disks allocated for the new virtual system.

### 1843 **9.3.1.2 Define virtual system with implementation-specific properties**

1844 **Assumption:** No assumption is made beyond the general assumption specified in 9.3.

- 1845 • The client performs use case 9.3.1.1 using an input configuration only. While preparing the input  
 1846 virtual system configuration, the client applies use case 9.2.14 to determine the implementation  
 1847 class of the CIM\_VirtualSystemSettingData class and use case 9.2.15 to determine the various  
 1848 implementation classes for the CIM\_ResourceAllocationSettingData class for the required  
 1849 resource types.

1850 The implementation classes may specify additional properties beyond the set that is defined in  
 1851 the respective base classes. The client may use the description information about each of these  
 1852 properties that is obtained with the respective class descriptions to request appropriate values  
 1853 from end users in order to create valid instances of the implementation class (thereby defining  
 1854 implementation-specific resource requirements).

1855 **Result:** The value of the DefinedSystem output parameter refers to an instance of the  
 1856 CIM\_ComputerSystem class that represents the newly created virtual system. The new system is in the  
 1857 "Defined" state.

## 1858 **9.3.2 Virtual system modification**

1859 This clause describes a set of usecases that modify virtual systems or virtual system configurations.

### 1860 **9.3.2.1 Modify virtual system state or definition**

1861 **Assumption:** The client knows a reference to an instance of the CIM\_ComputerSystem class that  
 1862 represents a virtual system.

1863 1) The client obtains the instance of the CIM\_VirtualSystemSettingData class that represents the  
 1864 state or definition of virtual aspects of the affected virtual system (respective use cases are de-  
 1865 scribed in [DSP1057](#)).

1866 2) The client makes conformant changes to the instance of the CIM\_VirtualSystemSettingData  
 1867 class. In particular, the client must not modify key properties.

1868 3) The client invokes the ModifySystemSettings( ) method (see 8.2.5) on the virtual system  
 1869 management service. The value of the SystemSettings parameter is the modified instance from  
 1870 step 2).

1871 4) The implementation executes the ModifySystemSettings( ) method, and the configuration of the  
 1872 virtual system is modified according to the clients requirements.

1873 **Result:** The requested modification is applied to the state or definition of the virtual system.

### 1874 **9.3.2.2 Add virtual resources**

1875 **Assumption:** The client knows a reference to an instance of the CIM\_VirtualSystemSettingData class  
 1876 that represents a virtual system configuration.

1877 1) The client locally prepares one or more instances of the CIM\_ResourceAllocationSettingData  
 1878 class to represent the resource allocation requests for the new virtual resources.

- 1879           2) The client invokes the AddResourceSettings( ) method (see 8.2.3) on the virtual system  
1880 management service, as follows:
- 1881           – The value of the AffectedConfiguration parameter is set to refer to the instance of the  
1882           CIM\_VirtualSystemSettingData class that represents the virtual system configuration that  
1883           receives new resources allocations.
  - 1884           – The value of the ResourceSettings[ ] array parameter is set with each element as one  
1885           embedded instance of the CIM\_ResourceAllocationSettingData class prepared in step 1).
- 1886           3) The implementation executes the AddResourceSettings( ) method, adding the requested re-  
1887           source allocations and resource allocation requests to the virtual system configuration.

1888           **Result:** The requested resource allocations or resource allocation requests are configured into the refer-  
1889           enced virtual system configuration.

### 1890   9.3.2.3   Modify virtual resource state extension or virtual resource definition

1891           **Assumption:** The client knows references to one or more instances of the CIM\_LogicalDevice class that  
1892           represent one or more virtual resources.

1893           Alternatively the client knows the reference to an instance of the CIM\_ResourceAllocationSettingData  
1894           class that represents the virtual resource state extensions or virtual resource definitions. In this case, the  
1895           client would obtain the referenced instance by using the intrinsic GetInstance( ) CIM operation and pro-  
1896           ceed with step 4).

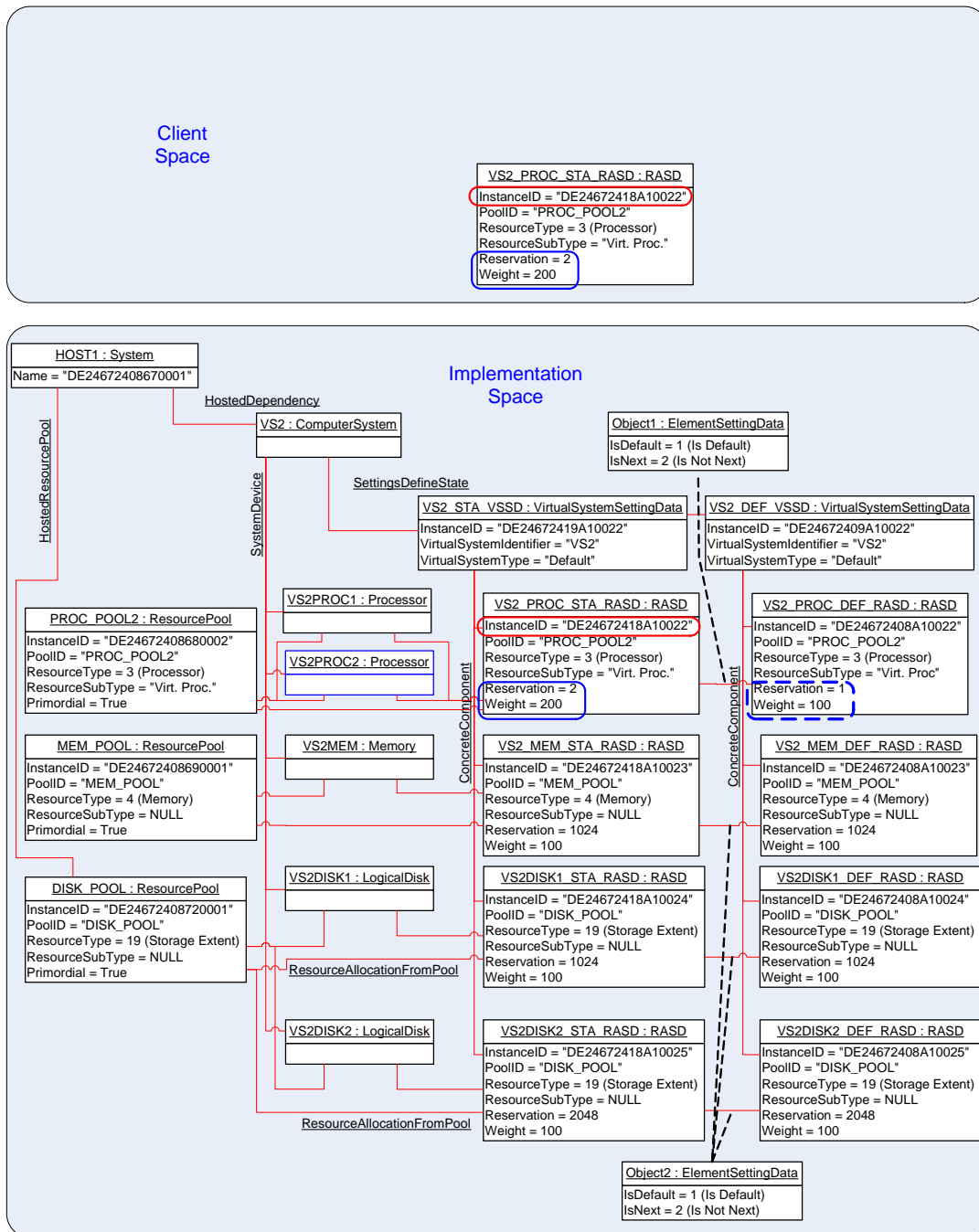
- 1897           1) The client invokes the intrinsic Associators( ) CIM operation for the virtual resource state exten-  
1898           sion as follows:
- 1899           – The value of the ObjectName parameter is set to refer to the instance of the  
1900           CIM\_LogicalDevice class.
  - 1901           – The value of the AssocClass parameter is set to "CIM\_SettingsDefineState".
  - 1902           – The value of the ResultClass parameter is set to "CIM\_ResourceAllocationSettingData".
- 1903           The result is a list of instances of the CIM\_ResourceAllocationSettingData class. The size of the  
1904           list is expected to be 1, and that element represents the virtual resource state extension. If the  
1905           client intends to modify the virtual resource state extension, the client skips steps 2) and 3), and  
1906           proceeds with step 4). If the client intends to modify the virtual resource definition, the client  
1907           continues with step 2).
- 1908           2) The client invokes the intrinsic References( ) CIM operation for the association instances that  
1909           connect the virtual resource definition, as follows:
- 1910           – The value of the ObjectName parameter is set to refer to the instance of the  
1911           CIM\_ResourceAllocationSettingData class that was obtained in step 1).
  - 1912           – The value of the ResultClass parameter is set to "CIM\_ElementSettingData".
- 1913           The result is a list of instances of the CIM\_ElementSettingData association that connect various  
1914           settings to the virtual resource state extension.
- 1915           3) The client selects from the result set of step 2) the instance in which the IsDefault property has  
1916           a value of 1 (Is Default). In that instance, the value of the SettingData property refers to the in-  
1917           stance of the CIM\_ResourceAllocationSettingData class that represents the virtual resource  
1918           definition.
- 1919           4) The client invokes the intrinsic GetInstance( ) CIM operation for the setting that represents the  
1920           resource allocation definition. The value of the InstanceName parameter is set to the value of  
1921           the SettingData property from the instance of the CIM\_ElementSettingData association selected  
1922           in step 3).
- 1923           The result is the instance of the CIM\_ResourceAllocationSettingData class that represents the  
1924           virtual resource definition.

- 1925 5) The client makes conformant changes to the instance of the  
1926 CIM\_ResourceAllocationSettingData class. In particular, the client must not modify key proper-  
1927 ties.
- 1928 Eventually the client executes steps 1) to 5) repetitively, preparing a set of resource allocation  
1929 change requests that subsequently are applied as one atomic operation.
- 1930 6) The client invokes the ModifyResourceSettings( ) method (see 8.2.4) on the virtual system man-  
1931 agement service. The values of elements of the ResourceSettings parameter are the modified  
1932 instances of the CIM\_ResourceAllocationSettingData class that were prepared through repeti-  
1933 tive execution of steps in steps 1) to 5).
- 1934 7) The implementation executes the ModifyResourceSettings( ) method, causing the requested re-  
1935 source allocation changes being applied to resource allocation state extensions or resource  
1936 allocation definitions.

1937 **Result:** The requested resource modifications are applied to virtual resource state extensions or virtual  
1938 resource definitions.

1939 Figure 5 shows the representation of a virtual system. Initially the virtual system was instantiated accord-  
1940 ing to the "Defined" virtual system configuration that is show on the right side. During the activation of the  
1941 virtual system, required resources were allocated. Virtual resources are represented by instances of sub-  
1942 classes of the CIM\_LogicalDevice class (CIM\_Processor, CIM\_Memory, or CIM\_LogicalDisk in this case),  
1943 with their "State" extensions in the "State" virtual system configuration. Related elements in the virtual  
1944 system representation and the "State" virtual system configuration are associated through instances of  
1945 the CIM\_SettingsDefineState association.

1946 Entities that are shown in blue color in Figure 5 are involved in the example of a processor resource  
1947 modification that is described following the figure.



1948

1949

**Figure 5 – Virtual system resource modification**

1950 Next, the client applied a resource modification on the allocated STA processor resource within the virtual system's "State" configuration. The "State" configuration is shown to the left of the "Defined" virtual system  
 1951 system's configuration. The client obtained a local copy of the instance of the `CIM_ResourceAllocationSettingData`  
 1952 class that is tagged `VS2_PROC_STA_RASD`. In that local copy, the client modified the value of the  
 1953 `Reservation` property to 2 and the value of the `Weight` property to 200. Then the client called the  
 1954 `ModifyResourceSettings()` method with the modified instance as the only element value for the  
 1955 `ResourceSettings[]` array parameter. The execution of that method resulted in another virtual processor  
 1956 being allocated to the virtual system.  
 1957

1958 NOTE: Because a change applied to the "State" virtual system configuration is temporary in nature, a recycling of  
 1959 the virtual system will nullify the change and result in a new "State" virtual system configuration based on the  
 1960 "Defined" virtual system configuration.

#### 1961 9.3.2.4 Delete virtual resources or virtual resource definitions

1962 **Assumption:** The client has references to one or more instances of the  
 1963 CIM\_ResourceAllocationSettingData class that refer to elements of the "State" or "Defined" virtual system  
 1964 configuration of one virtual system. See [DSP1057](#), clause 9, for respective use cases.

- 1965 1) The client invokes the RemoveResourceSettings( ) method (see 8.2.6) on the virtual system  
 1966 management service. The value of the ResourceSettings[ ] array parameter is set with each  
 1967 element referring to one instance of the CIM\_ResourceAllocationSettingData class.
- 1968 2) The implementation executes the RemoveResourceSettings( ) method. Either all requested re-  
 1969 source allocations or resource allocation requests are removed, or none at all.

1970 **Result:** The referenced virtual resources are removed from their respective virtual system configurations.

#### 1971 9.3.3 Destroy virtual system

1972 **Assumption:** The client knows a reference to an instance of the CIM\_ComputerSystem class that repre-  
 1973 sents a virtual system (see 9.2.16).

- 1974 1) The client invokes the DestroySystem( ) method on the virtual system management service.  
 1975 The value of the AffectedSystem parameter is set to refer to the instance of the  
 1976 CIM\_ComputerSystem class that represents the virtual system.
- 1977 2) The implementation executes the DestroySystem( ) method.

1978 **Result:** The affected virtual system and its virtual resources (together with their definition) are removed  
 1979 from the implementation. If the virtual system was in the "Active" state, the "Paused" state, or in the  
 1980 "Suspended" state, the running instance of the virtual system and its virtual resources are removed before  
 1981 the definition of the virtual system is removed.

1982 NOTE: Dependencies may exist that may prevent the destruction of a virtual system. For example, if definitions or  
 1983 instances of other virtual systems refer to elements of the virtual system to be destroyed, the destruction may fail.

### 1984 9.4 Snapshot-related activities

1985 This set of use cases describes activities such as the following:

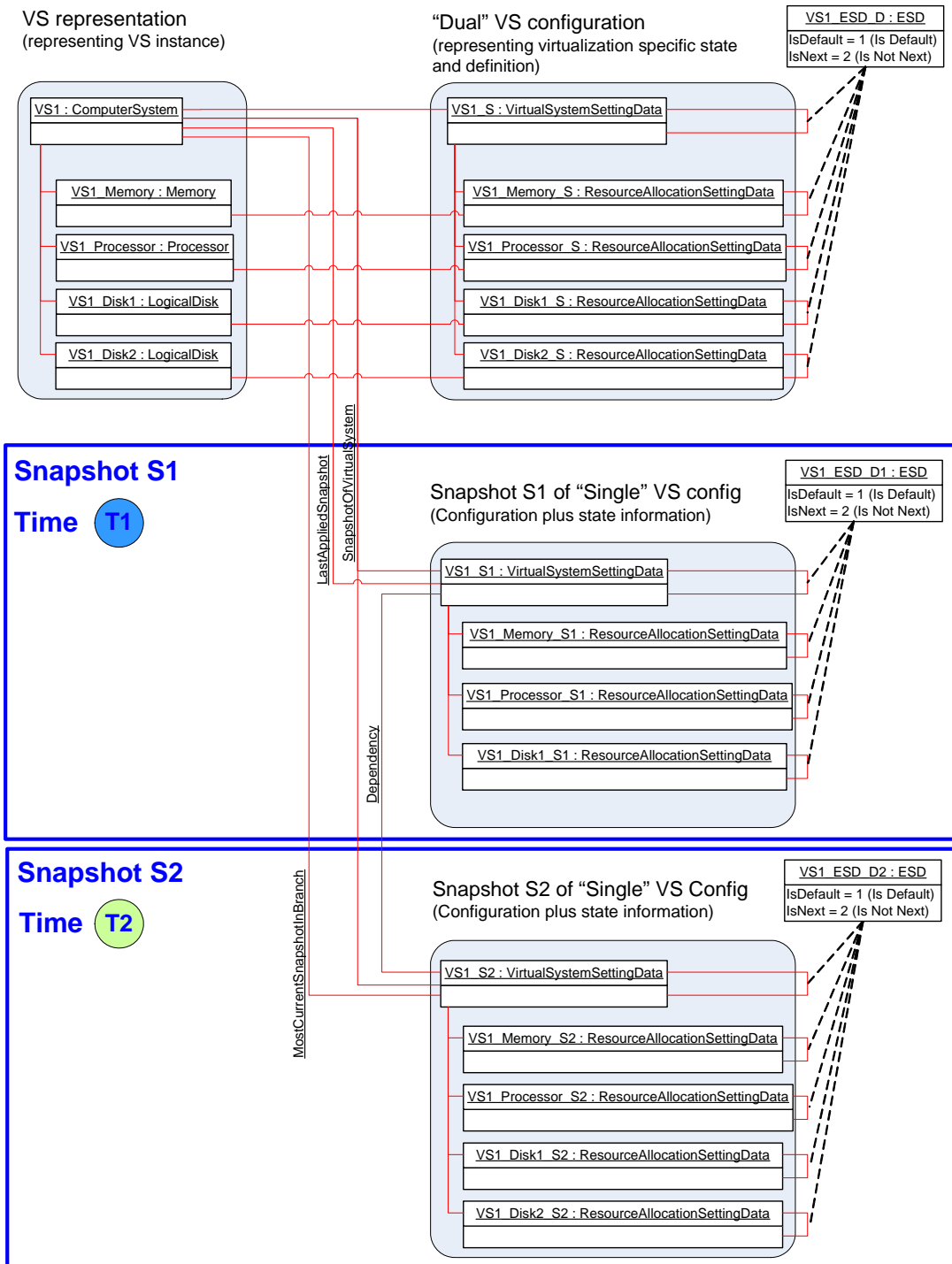
- 1986 • discovering a virtual system snapshot service
- 1987 • inspecting the capabilities of a virtual system snapshot service
- 1988 • creating a snapshot from a virtual system
- 1989 • applying a snapshot to a virtual system
- 1990 • analyzing a virtual snapshot
- 1991 • analyzing dependencies among snapshots
- 1992 • locating the most recently captured snapshot
- 1993 • destroying a snapshot

1994 Figure 6 depicts the CIM representation of a virtual system VS1 and of configurations that are associated  
 1995 with the virtual system at time T3. In the example, it is assumed that the implementation applies the  
 1996 "Single-Configuration Implementation Approach" as described in [DSP1057](#).

1997 The sequence of events that yield the situation shown in Figure 6 is as follows:

- 1998  
1999
- 2000
- 2001  
2002  
2003
- 2004  
2005
- 2006  
2007
- 2008  
2009  
2010  
2011
- 2012  
2013
- 2014  
2015
- 2016  
2017  
2018
- 2019
- 2020  
2021  
2022
- 1) At time T0, the virtual system VS1 is defined. The initial virtual system definition contains virtual resource allocation requests for one memory extent, one virtual processor, and one virtual disk.
  - 2) At a time after T0 but before T1, the virtual system is activated.
  - 3) At time T1, a full snapshot S1 is captured of the virtual system. Virtual system definition and state are copied into the snapshot. A full snapshot includes the "content" of virtual memory *and* of virtual disks; a disk snapshot would contain the "content" of virtual disks only.
  - 4) The virtual system remains active after the snapshot is captured. The virtual system configuration and the "content" of memory and of virtual disks may change in that interval.
  - 5) At a time after T1 but before T2, snapshot S1 is applied to the virtual system, causing definition and state to be restored to the situation at time T1.
  - 6) Still at a time before T2, a second virtual disk is dynamically added to the virtual system. Because in this example the implementation applies the "Single-Configuration Implementation Approach," this change in effect applies to both virtual system definition and virtual system instance and is visible through the "Single" VS configuration.
  - 7) At time T2, snapshot S2 is captured of the virtual system. Because at time T2 the virtual system snapshot S1 is the last applied snapshot, snapshot S2 depends on snapshot S1.
  - 8) The virtual system remains active after the snapshot is captured. The virtual system configuration and the "content" of memory and of virtual disks may change in that interval.
  - 9) At a time after T2 but before T3, snapshot S2 is applied to the virtual system, causing definition and state to be restored to the situation at time T2, thereby nullifying changes that were applied to the virtual system after T2.
  - 10) At time T3, the situation is as shown in Figure 6.
- General assumption:** The client knows the reference to an instance of the CIM\_VirtualSystemSnapshotService class that represents the virtual system snapshot of a host system (see 9.2.6).

**Current Time: T3 > T2 > T1 > T0**



2023

2024

**Figure 6 – System Virtualization Profile: Snapshot example**

### 2025 9.4.1 Locate virtual system snapshot service

2026 **Assumption:** The client knows a reference to an instance of the CIM\_System class that represents a  
2027 host system that is a central instance of this profile; see 9.2.4.

- 2028
- The client invokes the intrinsic AssociatorNames( ) CIM operation as follows:
    - 2029 – The value of the ObjectName parameter is set to refer to the instance of the CIM\_System
    - 2030 class.
    - 2031 – The value of the AssocClass parameter is set to "CIM\_HostedService".
    - 2032 – The value of the ResultClass parameter is set to "CIM\_VirtualSystemSnapshotService".

2033 **Result:** The client knows a reference to the instance of the CIM\_VirtualSystemSnapshotService class  
2034 that represents the virtual system snapshot service serving the host system. If the operation is successful,  
2035 the size of the result set is 1.

2036 In the example shown in Figure 3, one instance of the CIM\_VirtualSystemSnapshotService class serves  
2037 the host system; it is tagged VSSS\_1.

### 2038 9.4.2 Determine capabilities of a virtual system snapshot service

2039 **Assumption:** The client knows a reference to an instance of the CIM\_VirtualSystemSnapshotService  
2040 class that represents the virtual system snapshot service serving a host system (see 9.4.1).

- 2041 1) The client invokes the intrinsic Associators( ) CIM operation as follows:
- 2042 – The value of the ObjectName parameter is set to refer to the instance of the
  - 2043 CIM\_VirtualSystemSnapshotService class.
  - 2044 – The value of the AssocClass parameter is set to "CIM\_ElementCapabilities".
  - 2045 – The value of the ResultClass parameter is set to
  - 2046 "CIM\_VirtualSystemSnapshotServiceCapabilities".
- 2047 The result is a list of instances of the CIM\_VirtualSystemSnapshotServiceCapabilities class. If  
2048 the operation is successful, the size of the result set is 1.
- 2049 2) The client analyzes the instance of the CIM\_VirtualSystemSnapshotServiceCapabilities class.
- 2050 – The SynchronousMethodsSupported[ ] array property lists identifiers of methods of the
  - 2051 CIM\_VirtualSystemSnapshotServiceCapabilities class that are implemented with
  - 2052 synchronous method execution only.
  - 2053 – The AsynchronousMethodsSupported[ ] array property lists identifiers of methods of the
  - 2054 CIM\_VirtualSystemSnapshotServiceCapabilities class that are implemented with
  - 2055 synchronous and asynchronous method execution.
  - 2056 – The SnapshotTypesSupported[ ] array property lists identifiers designating snapshot types
  - 2057 that are supported by the implementation.

2058 **Result:** The client knows the virtual-system-snapshot-related capabilities of the host system in terms of  
2059 properties of the CIM\_VirtualSystemSnapshotServiceCapabilities class.

2060 In the example shown in Figure 3, one instance of the CIM\_VirtualSystemSnapshotServiceCapabilities  
2061 class is associated with the host system; it is tagged VSSSC\_1.



### 2062 9.4.3 Create snapshot

2063 **Assumption:** The client knows a reference to an instance of the CIM\_ComputerSystem class that repre-  
2064 sents a virtual system hosted by a host system (see 9.2.16). The virtual system is active.

2065 1) The client invokes the CreateSnapshot( ) method on the virtual system snapshot service, as fol-  
2066 lows:

2067 – The value of the AffectedSystem parameter is set to refer to the instance of the  
2068 CIM\_ComputerSystem class that represents the virtual system.

2069 – The value of the SnapshotType parameter is set to 2 (Full Snapshot).

2070 2) The implementation executes the CreateSnapshot( ) method.

2071 The value returned in the ResultingSnapshot parameter refers to an instance of the  
2072 CIM\_VirtualSystemSettingData class that represents the new snapshot.

2073 **Result:** The client knows a reference to the instance of the CIM\_VirtualSystemSettingData class that  
2074 represents the created virtual system snapshot.

2075 In the example shown in Figure 6, two instances of the CIM\_VirtualSystemSettingData class represent  
2076 virtual system snapshots S1 and S2 taken at times T1 and T2. Although the situation captured in Figure 6  
2077 shows the situation at T3, a snapshot taken at T3 would look identical to S2 (because the current system  
2078 at time T3 is unchanged with respect to S2).

### 2079 9.4.4 Locate snapshots of a virtual system

2080 **Assumption:** The client knows a reference to an instance of the CIM\_ComputerSystem class that  
2081 represents a virtual system (see 9.2.16).

2082 • The client invokes the intrinsic Associators( ) CIM operation for the list of snapshots, as follows:

2083 – The value of the ObjectName parameter is set to refer to the instance of the  
2084 CIM\_ComputerSystem class.

2085 – The value of the AssocClass parameter is set to "CIM\_SnapshotOfVirtualSystem".

2086 – The value of the ResultClass parameter is set to "CIM\_VirtualSystemSettingData".

2087 The result is a list of instances of the CIM\_VirtualSystemSettingData class.

2088 **Result:** The client knows a set of instances of the CIM\_VirtualSystemSettingData class, each represent-  
2089 ing a virtual system snapshot taken from the virtual system.

2090 In the example shown in Figure 6, the instances tagged VS\_S1 and VS1\_S2 of the  
2091 CIM\_VirtualSystemSettingData class represent snapshots S1 and S2.

### 2092 9.4.5 Locate the source virtual system of a snapshot

2093 **Assumption:** The client knows the reference to an instance of the CIM\_VirtualSystemSettingData class  
2094 that represents a virtual system snapshot.

2095 • The client invokes the intrinsic AssociatorNames( ) CIM operation for the source virtual system  
2096 as follows:

2097 – The value of the ObjectName parameter is set to refer to the instance of the  
2098 CIM\_VirtualSystemSettingData class.

2099 – The value of the AssocClass parameter is set to "CIM\_ElementSettingData".

2100 – The value of the ResultClass parameter is set to "CIM\_ComputerSystem".

2101 The result is a list of references to instances of the CIM\_ComputerSystem class. The size of the  
2102 list is 1.

2103 **Result:** The client knows a reference to an instance of the CIM\_ComputerSystem class that represents  
2104 the virtual system that was the source for the snapshot.

2105 NOTE: At this time the present configuration of the virtual system may be completely different from the configuration  
2106 that was captured in the snapshot.

2107 In the example shown in Figure 6, the instance of class CIM\_ComputerSystem tagged VS1 is the source  
2108 of snapshots S1 and S2, represented by instances of the CIM\_VirtualSystemSettingData class tagged  
2109 VS\_S1 and VS\_S2.

#### 2110 9.4.6 Locate the most current snapshot in a branch of snapshots

2111 **Assumption:** The client knows an instance of the CIM\_ComputerSystem class that represents a virtual  
2112 system (see 9.2.16).

- 2113 • The client invokes the intrinsic Associators( ) CIM operation for the most current snapshot in the  
2114 current branch of virtual snapshots, as follows:
    - 2115 – The value of the ObjectName parameter is set to refer to the instance of the  
2116 CIM\_ComputerSystem class.
    - 2117 – The value of the AssocClass parameter is set to "CIM\_MostCurrentSnapshotInBranch".
    - 2118 – The value of the ResultClass parameter is set to "CIM\_VirtualSystemSettingData".
- 2119 The result is a list of instances of the CIM\_VirtualSystemSettingData class. The size of the list is  
2120 1.

2121 **Result:** The client knows an instance of the CIM\_VirtualSystemSettingData class that represents the vir-  
2122 tual system snapshot that is the most current snapshot in the current branch of snapshots.

2123 In the example shown in Figure 6, the instance of the CIM\_VirtualSystemSettingData class that is tagged  
2124 VS1\_2 represents the most current snapshot in the current branch of snapshots. This is the case because  
2125 that snapshot was applied most recently to the virtual system and no other snapshot was applied to or  
2126 created from the virtual system since then.

#### 2127 9.4.7 Locate dependent snapshots

2128 **Assumption:** The client knows a reference to an instance of the CIM\_VirtualSystemSettingData class  
2129 that represents a virtual system snapshot (see 9.4.4).

- 2130 • The client invokes the intrinsic AssociatorNames( ) CIM operation for the list of dependent snap-  
2131 shots as follows:
    - 2132 – The value of the ObjectName parameter is set to refer to the instance of the  
2133 CIM\_VirtualSystemSettingData class.
    - 2134 – The value of the AssocClass parameter is set to "CIM\_Dependency".
    - 2135 – The value of the ResultClass parameter is set to "CIM\_VirtualSystemSettingData".
    - 2136 – The value of the Role parameter is set to "Antecedent".
    - 2137 – The value of the ResultRole parameter is set to "Dependent".
- 2138 The result is a list of references to instances of the CIM\_VirtualSystemSettingData class.

2139 **Result:** The client knows a set of instances of the CIM\_VirtualSystemSettingData class that represent vir-  
2140 tual system snapshots that depend on the input virtual system snapshot. The set may be empty, indicating  
2141 that no dependent snapshots exist.

2142 In the example shown in Figure 6, the instance tagged VS\_S2 represents snapshot S2, which is depend-  
2143 ent on snapshot S1, which is represented by the instance tagged VS\_S1.

### 2144 9.4.8 Locate parent snapshot

2145 **Assumption:** The client knows a reference to an instance of the CIM\_VirtualSystemSettingData class  
2146 that represents a virtual system snapshot (see 9.4.4).

2147 • The client invokes the intrinsic AssociatorNames( ) CIM operation for the parent snapshot as fol-  
2148 lows:

2149 – The value of the ObjectName parameter is set to refer to the instance of the  
2150 CIM\_VirtualSystemSettingData class that represents the virtual system snapshot.

2151 – The value of the AssocClass parameter is set to "CIM\_Dependency".

2152 – The value of the ResultClass parameter is set to "CIM\_VirtualSystemSettingData".

2153 – The value of the Role parameter is set to "Dependent".

2154 – The value of the ResultRole parameter is set to "Antecedent".

2155 The result is a list of references to instances of the CIM\_VirtualSystemSettingData class that  
2156 represent virtual system snapshots. The list has a size of 1 or 0.

2157 **Result:** The client knows the instance of the CIM\_VirtualSystemSettingData class that represents the par-  
2158 ent virtual system snapshot of the input virtual system snapshot. The set may be empty, indicating that no  
2159 parent snapshots exist.

2160 In the example shown in Figure 6, the instance tagged VS\_S1 represents snapshot S1, which is the par-  
2161 ent of snapshot S2, which is represented by the instance tagged VS\_S2.

### 2162 9.4.9 Apply snapshot

2163 **Assumption:** The client knows a reference to an instance of the CIM\_VirtualSystemSettingData class  
2164 that represents a virtual system snapshot (see 9.4.3 or 9.4.4). The client knows a reference to the in-  
2165 stance of the CIM\_ComputerSystem class that represents the virtual system that was the source for the  
2166 snapshot (see 9.4.5). The virtual system is active.

2167 1) The client invokes the ApplySnapshot( ) method on the virtual system snapshot service. The  
2168 value of the Snapshot parameter is set to refer to the instance of the  
2169 CIM\_VirtualSystemSettingData class that represents the snapshot.

2170 2) The snapshot is applied into the active virtual system as follows:

2171 a) The virtual system is deactivated. This implies a disruptive termination of the software that  
2172 may be active in the instance of the virtual system.

2173 b) The virtual system is reconfigured according to the virtual system snapshot. For a disk  
2174 snapshot, this applies to the disk resources only.

2175 c) If the applied snapshot is a full snapshot, all stateful resources like memory and disk are  
2176 restored to the situation that was captured in the snapshot. If the applied snapshot is a disk  
2177 snapshot, only disk resources are restored.

2178 d) The virtual system is activated. If the applied snapshot is a full snapshot, the virtual system  
2179 starts from the situation that was captured by the full snapshot. If the applied snapshot was  
2180 a disk snapshot, a normal virtual system activation occurs.

2181 **Result:** The virtual system is restored to the situation that was in place when the snapshot was taken.

2182 In the example shown in Figure 6, the situation is depicted at time T3, immediately after the activation of  
2183 snapshot S2 within virtual system VS1.

2184 **9.4.10 Destroy snapshot**

2185 **Assumption:** The client knows the reference to an instance of the CIM\_VirtualSystemSettingData class  
2186 that represents a virtual system snapshot (see 9.2.16).

- 2187 1) The client invokes the DestroySnapshot( ) method on the virtual system management service.  
2188 The value of the Snapshot parameter is set to refer to the instance of the  
2189 CIM\_VirtualSystemSettingData class that represents the snapshot.
- 2190 2) The snapshot is removed from the implementation.

2191 **Result:** The snapshot no longer exists within the implementation.

2192 **10 CIM elements**

2193 Table 20 lists CIM elements that are defined or specialized for this profile. Each CIM element shall be  
2194 implemented as described in Table 20. The CIM Schema descriptions for any referenced element and its  
2195 sub-elements apply.

2196 Clauses 7 ("Implementation") and 8 ("Methods") may impose additional requirements on these elements.

2197 **Table 20 – CIM Elements: System Virtualization Profile**

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

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

2198 **10.1 CIM\_AffectedJobElement**

2199 The implementation of the CIM\_AffectedJobElement association is conditional.

2200 Condition: A non-NULL value for at least one element of the AsynchronousMethodsSupported[ ] array  
 2201 property of the CIM\_VirtualSystemManagementCapabilities class is implemented.

2202 If the CIM\_AffectedJobElement association is implemented, the provisions in this subclause apply.

2203 An implementation shall use the CIM\_AffectedJobElement association to associate an instance of the  
 2204 CIM\_ConcreteJob class that represents an asynchronous task and an instance of the  
 2205 CIM\_ComputerSystem class that represents a virtual system that is affected by its execution.

2206 Table 21 contains the requirements for elements of this association.

2207 **Table 21 – Association: CIM\_AffectedJobElement**

| Elements          | Requirement | Notes   |
|-------------------|-------------|---|
| AffectedElement   | Mandatory   | <b>Key:</b> See 8.1.2.<br><b>Cardinality:</b> * |
| AffectingElement  | Mandatory   | <b>Key:</b> See 8.1.2.<br><b>Cardinality:</b> 1 |
| ElementEffects[ ] | Mandatory   | See 8.1.2.                                      |

2208 **10.2 CIM\_ConcreteJob**

2209 The implementation of the CIM\_ConcreteJob class is conditional.

2210 Condition: A non-NULL value for at least one element of the AsynchronousMethodsSupported[ ] array  
 2211 property of the CIM\_VirtualSystemManagementCapabilities class is implemented.

2212 If the CIM\_ConcreteJob class is implemented, the provisions in this subclause apply.

2213 An implementation shall use an instance of the CIM\_ConcreteJob class to represent an asynchronous  
 2214 task.

2215 Table 22 contains requirements for elements of this class.

2216 **Table 22 – Class: CIM\_ConcreteJob**

| Elements              | Requirement | Notes      |
|-----------------------|-------------|------------|
| InstanceID            | Mandatory   | <b>Key</b> |
| JobState              | Mandatory   | See 8.1.2. |
| TimeOfLastStateChange | Mandatory   | See 8.1.2. |

### 2217 10.3 CIM\_Dependency

2218 The implementation of the CIM\_Dependency association is conditional.

2219 Condition: Virtual system snapshots are implemented; see 7.7.1.1.

2220 If the CIM\_Dependency association class is implemented, the provisions in this subclause apply.

2221 An implementation shall use an instance of the CIM\_Dependency association to associate an instance of  
2222 the CIM\_VirtualSystemSettingData class that represents a parent snapshot and an instance of the  
2223 CIM\_VirtualSystemSettingData class that represents a dependent snapshot.

2224 Table 23 contains requirements for elements of this class.

2225 **Table 23 – Class: CIM\_Dependency Class**

| Elements   | Requirement | Notes   |
|------------|-------------|---|
| Antecedent | Mandatory   | <b>Key:</b> Reference to an instance of the CIM_VirtualSystemSettingData class that represents a parent snapshot<br><b>Cardinality:</b> 0..1    |
| Dependent  | Mandatory   | <b>Key:</b> Reference to an instance of the CIM_VirtualSystemSettingData class that represents a dependent snapshot<br><b>Cardinality:</b> 0..1 |

### 2226 10.4 CIM\_ElementCapabilities (Host system)

2227 An implementation shall use an instance of the CIM\_ElementCapabilities association to associate an in-  
2228 stance of the CIM\_System class that represents a host system with an instance of the  
2229 CIM\_VirtualSystemManagementCapabilities class that describes the virtual system management capabili-  
2230 ties of the host system.

2231 Table 24 contains requirements for elements of this association.

2232 **Table 24 – Association: CIM\_ElementCapabilities (Host System)**

| Elements       | Requirement | Notes   |
|----------------|-------------|---|
| ManagedElement | Mandatory   | <b>Key:</b> Reference to instance of the CIM_System class that represents a host system<br><b>Cardinality:</b> 1  |
| Capabilities   | Mandatory   | <b>Key:</b> Reference to an instance of the CIM_VirtualSystemManagementCapabilities class that describes the capabilities of a host system<br><b>Cardinality:</b> 1 |

### 2233 10.5 CIM\_ElementCapabilities (Virtual system management service)

2234 The implementation of the CIM\_ElementCapabilities association for the virtual system management  
2235 service is conditional.

2236 Condition: Any of the following is implemented:

- 2237 • Virtual system definition and destruction (see 7.4.6.1)

- 2238 • Virtual resource addition and removal (see 7.4.6.2)
- 2239 • Virtual system and resource modification (see 7.4.6.3)

2240 If the CIM\_ElementCapabilities association is implemented for the virtual system management service,  
2241 the provisions in this subclause apply.

2242 An implementation shall use an instance of the CIM\_ElementCapabilities association to associate an in-  
2243 stance of the CIM\_VirtualSystemManagementService class that represents a virtual system management  
2244 service with an instance of the CIM\_VirtualSystemManagementCapabilities that describes the capabilities  
2245 of the virtual system management service.

2246 Table 25 contains requirements for elements of this association.

2247 **Table 25 – Association: CIM\_ElementCapabilities (Virtual system management)**

| Elements       | Requirement | Notes  |
|----------------|-------------|--|
| ManagedElement | Mandatory   | <b>Key:</b> Reference to instance of the CIM_VirtualSystemManagementService class<br><b>Cardinality:</b> 0..1      |
| Capabilities   | Mandatory   | <b>Key:</b> Reference to an instance of the CIM_VirtualSystemManagementCapabilities class<br><b>Cardinality:</b> 1 |

2248 **10.6 CIM\_ElementCapabilities (Virtual system snapshot service)**

2249 The implementation of the CIM\_ElementCapabilities association for the virtual system snapshot service is  
2250 conditional.

2251 Condition: Virtual system snapshots are implemented; see 7.7.1.1.

2252 If the CIM\_ElementCapabilities association is implemented for the virtual system snapshot service, the  
2253 provisions in this subclause apply.

2254 An implementation shall use an instance of the CIM\_ElementCapabilities association to associate an in-  
2255 stance of the CIM\_VirtualSystemSnapshotService class that represents a virtual system snapshot service  
2256 with an instance of the CIM\_VirtualSystemSnapshotServiceCapabilities class that describes the capabili-  
2257 ties of the virtual system snapshot service.

2258 Table 26 contains requirements for elements of this association.

2259 **Table 26 – Association: CIM\_ElementCapabilities (Snapshot service)**

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

## 2260 10.7 CIM\_ElementCapabilities (Snapshots of virtual systems)

2261 The implementation of the CIM\_ElementCapabilities association for the virtual systems snapshots is  
2262 conditional.

2263 Condition: Virtual system snapshots are implemented; see 7.7.1.1.

2264 If the CIM\_ElementCapabilities association is implemented for virtual systems snapshots, the provisions  
2265 in this subclause apply.

2266 The implementation shall use an instance of the CIM\_ElementCapabilities association to associate in-  
2267 stances of the CIM\_VirtualSystemSnapshotCapabilities class with those instances of the  
2268 CIM\_ComputerSystem class that represent a virtual system to which the capabilities apply.

2269 Table 27 contains requirements for elements of this association.

2270 **Table 27 – Association: CIM\_ElementCapabilities (Snapshots of virtual systems)**

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

## 2271 10.8 CIM\_ElementConformsToProfile

2272 An implementation shall use an instance of the CIM\_ElementConformsToProfile association to associate  
2273 an instance of the CIM\_RegisteredProfile class that represents an implementation of this profile with  
2274 instances of the CIM\_System class that represent a host system that is a central and scoping instance of  
2275 this profile.

2276 Table 28 contains requirements for elements of this association.

2277 **Table 28 – Association: CIM\_ElementConformsToProfile**

| Elements           | Requirement | Notes  |
|--------------------|-------------|--|
| ConformantStandard | Mandatory   | <b>Key:</b> Reference to an instance of the CIM_RegisteredProfile class that represents an implementation of this profile<br><b>Cardinality:</b> 1 |
| ManagedElement     | Mandatory   | <b>Key:</b> Reference to an instance of the CIM_System class that represents a host system<br><b>Cardinality:</b> *                                |



2278 **10.9 CIM\_HostedDependency**

2279 An implementation shall use an instance of the CIM\_HostedDependency association to associate an  
 2280 instance of the CIM\_System class that represents a host system with each instance of the CIM\_Comput-  
 2281 erSystem class that represents a virtual system hosted by the host system.

2282 Table 29 contains requirements for elements of this association.

2283 **Table 29 – Association: CIM\_HostedDependency**

| Elements   | Requirement | Notes  |
|------------|-------------|--|
| Antecedent | Mandatory   | <b>Key:</b> Reference to an instance of the CIM_System class that represents a host system<br><b>Cardinality:</b> 1            |
| Dependent  | Mandatory   | <b>Key:</b> Reference to an instance of the CIM_ComputerSystem class that represents a virtual system<br><b>Cardinality:</b> * |

2284 **10.10 CIM\_HostedService (Virtual system management service)**

2285 The implementation of the CIM\_HostedService association for the virtual system management service is  
 2286 conditional:

2287 Condition: Any of the following is implemented:

- 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 If the CIM\_HostedService association is implemented for the virtual system management service, the  
 2292 provisions in this subclause apply.

2293 The implementation shall use an instance of the CIM\_HostedService association to associate an instance  
 2294 of the CIM\_System class that represents a host system and the instance of the CIM\_VirtualSystem-  
 2295 ManagementService class that represents the virtual system management service that is hosted by a  
 2296 host system.

2297 Table 30 contains requirements for elements of this association.

2298 **Table 30 – Association: CIM\_HostedService (Virtual system management service)**

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

## 2299 10.11 CIM\_HostedService (Virtual system snapshot service)

2300 The implementation of the CIM\_HostedService association is conditional.

2301 Condition: Virtual system snapshots are implemented; see 7.7.1.1.

2302 If the CIM\_HostedService association is implemented for the virtual system snapshot service, the  
2303 provisions in this subclause apply.

2304 The implementation shall use an instance of the CIM\_HostedService association to associate an instance  
2305 of the CIM\_ComputerSystem class that represents a host system and the instance of the  
2306 CIM\_VirtualSystemSnapshotService class that represents the virtual system snapshot service.

2307 Table 31 contains requirements for elements of this association.

2308 **Table 31 – Association: CIM\_HostedService (Virtual system snapshot service)**

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

## 2309 10.12 CIM\_LastAppliedSnapshot

2310 The implementation of the CIM\_LastAppliedSnapshot association is conditional.

2311 Condition: Virtual system snapshots are implemented; see 7.7.1.1.

2312 If the CIM\_LastAppliedSnapshot association is implemented, the provisions in this subclause apply.

2313 An implementation shall use an instance of the CIM\_LastAppliedSnapshot association to associate an in-  
2314 stance of the CIM\_ComputerSystem class that represents a virtual system and the instance of the  
2315 CIM\_VirtualSystemSettingData class that represents the virtual system snapshot that was last applied to  
2316 the virtual system.

2317 Table 32 contains requirements for elements of this association.

2318 **Table 32 – Association: CIM\_LastAppliedSnapshot**

| Elements   | Requirement | Notes  |
|------------|-------------|--|
| Antecedent | Mandatory   | <b>Key:</b> Reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot<br><b>Cardinality:</b> 0..1 |
| Dependent  | Mandatory   | <b>Key:</b> Reference to the instance of the CIM_ComputerSystem class that represents the virtual system<br><b>Cardinality:</b> 0..1                 |

2319 **10.13 CIM\_MostCurrentSnapshotInBranch**

2320 The implementation of the CIM\_MostCurrentSnapshotInBranch association is conditional.

2321 Condition: Virtual system snapshots are implemented; see 7.7.1.1.

2322 If the CIM\_MostCurrentSnapshotInBranch association is implemented, the provisions in this subclause  
2323 apply.

2324 An implementation shall use an instance of the CIM\_MostCurrentSnapshotInBranch association to  
2325 associate an instance of the CIM\_ComputerSystem class that represents a virtual system and the  
2326 instance of the CIM\_VirtualSystemSettingData class that represents the most current snapshot in a  
2327 branch of virtual system snapshots. The most current snapshot in a branch of snapshots related to an in-  
2328 stance of a virtual system is the younger of the following snapshots:

- 2329 • the snapshot that was most recently captured from the virtual system instance
- 2330 • the snapshot that was last applied to the instance

2331 Table 33 contains requirements for elements of this association.

2332 **Table 33 – Association: CIM\_MostCurrentSnapshotInBranch**

| Elements   | Requirement | Notes  |
|------------|-------------|--|
| Antecedent | Mandatory   | <b>Key:</b> Reference to the instance of the CIM_ComputerSystem class that represents the virtual system<br><b>Cardinality:</b> 0..1                 |
| Dependent  | Mandatory   | <b>Key:</b> Reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot<br><b>Cardinality:</b> 0..1 |

2333 **10.14 CIM\_ReferencedProfile**

2334 The implementation of the CIM\_ReferencedProfile association is conditional.

2335 Condition: Resource virtualization profiles such as [DSP1059](#) are implemented as scoped profiles.

2336 If the CIM\_ReferencedProfile association is implemented, the provisions in this subclause apply.

2337 An implementation shall use an instance of the CIM\_ReferencedProfile association to associate an in-  
2338 stance of the CIM\_RegisteredProfile class that represents an implementation of this profile and any  
2339 instance of the CIM\_RegisteredProfile class that represents an implementation of a resource allocation  
2340 DMTF management profile that describes virtual resource allocation that is implemented by the  
2341 implementation.

2342 Table 34 contains requirements for elements of this association.

2343 **Table 34 – Association: CIM\_ReferencedProfile**

| Elements   | Requirement | Notes   |
|------------|-------------|---|
| Antecedent | Mandatory   | <b>Key:</b> Reference to an instance of the CIM_RegisteredProfile that represents an implementa-<br>tion of this profile<br><b>Cardinality:</b> 1 |

| Elements  | Requirement | Notes   |
|-----------|-------------|---|
| Dependent | Mandatory   | <b>Key:</b> Reference to an instance of the CIM_RegisteredProfile class that represents an implementation of a resource allocation profile<br><b>Cardinality:</b> * |

## 2344 10.15 CIM\_RegisteredProfile

2345 An implementation shall use an instance of the CIM\_RegisteredProfile class to represent an  
2346 implementation of this profile.

2347 Table 35 contains requirements for elements of this class.

2348 **Table 35 – Class: CIM\_RegisteredProfile**

| Elements               | Requirement | Notes  |
|------------------------|-------------|--|
| InstanceID             | Mandatory   | <b>Key</b>   |
| 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"). |

## 2349 10.16 CIM\_ServiceAffectsElement (Virtual system management service)

2350 The implementation of the CIM\_ServiceAffectsElement association for the virtual system management  
2351 service is conditional.

2352 Condition: Any of the following is implemented:

- 2353 • Virtual system definition and destruction (see 7.4.6.1)
- 2354 • Virtual resource addition and removal (see 7.4.6.2)
- 2355 • Virtual system and resource modification (see 7.4.6.3)

2356 If the CIM\_ServiceAffectsElement association is implemented for the virtual system management service,  
2357 the provisions in this subclause apply.

2358 The implementation shall use an instance of the CIM\_ServiceAffectsElement association to associate an  
2359 instance of the CIM\_VirtualSystemManagementService class that represents a virtual system manage-  
2360 ment service and any instance of the CIM\_ComputerSystem class that represents a virtual system that is  
2361 managed by that virtual system management service.

2362 Table 36 contains requirements for elements of this association.

2363 **Table 36 – Association: CIM\_ServiceAffectsElement (Virtual system management service)**

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

2364 **10.17 CIM\_ServiceAffectsElement (Virtual system snapshot service)**

2365 The implementation of the CIM\_ServiceAffectsElement association is conditional.

2366 Condition: Virtual system snapshots are implemented; see 7.7.1.1.

2367 If the CIM\_ServiceAffectsElement association is implemented for the virtual system snapshot service, the  
2368 provisions in this subclause apply.

2369 The implementation shall use an instance of the CIM\_ServiceAffectsElement association to associate an  
2370 instance of the CIM\_VirtualSystemSnapshotService class that represents a virtual system management  
2371 service with the following instances:

- 2372 • any instance of the CIM\_ComputerSystem class that represents a virtual system that is man-  
2373 aged by that virtual system management service
- 2374 • any instance of the CIM\_VirtualSystemSettingData class that represents a virtual system snap-  
2375 shot

2376 Table 37 contains requirements for elements of this association.

2377 **Table 37 – Association: CIM\_ServiceAffectsElement**

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

2378 **10.18 CIM\_SnapshotOfVirtualSystem**

2379 The implementation of the CIM\_SnapshotOfVirtualSystem association is conditional.

2380 Condition: Virtual system snapshots are implemented; see 7.7.1.1.

2381 If the CIM\_SnapshotOfVirtualSystem association is implemented, the provisions in this subclause apply.

2382 An implementation shall use an instance of the CIM\_SnapshotOfVirtualSystem association to associate  
2383 an the instance of the CIM\_ComputerSystem class that represents the virtual system that was the source

2384 for the virtual system snapshot and the instance of the CIM\_VirtualSystemSettingData class that repre-  
2385 sents a snapshot of the virtual system

2386 Table 38 contains requirements for elements of this association.

2387 **Table 38 – Association: CIM\_SnapshotOfVirtualSystem**

| Elements   | Requirement | Notes   |
|------------|-------------|---|
| Antecedent | Mandatory   | <b>Key:</b> Reference to the instance of the CIM_Computer-System class that represents the source virtual system<br><b>Cardinality:</b> 0..1      |
| Dependent  | Mandatory   | <b>Key:</b> Reference to an instance of the CIM_VirtualSystemSettingData class that represents a virtual system snapshot<br><b>Cardinality:</b> * |

## 2388 10.19 CIM\_System

2389 An implementation shall use an instance of a concrete subclass of the CIM\_System class to represent a  
2390 host system.

2391 Table 39 contains requirements for elements of this class.

2392 **Table 39 – Class: CIM\_VirtualSystemManagementCapabilities**

| Elements          | Requirement | Notes      |
|-------------------|-------------|------------|
| CreationClassName | Mandatory   | <b>Key</b> |
| Name              | Mandatory   | <b>Key</b> |

## 2393 10.20 CIM\_VirtualSystemManagementCapabilities

2394 An implementation shall use an instance of the CIM\_VirtualSystemManagementCapabilities class to  
2395 represent the virtual system management capabilities of a host system.

2396 Table 40 contains requirements for elements of this class.

2397 **Table 40 – Class: CIM\_VirtualSystemManagementCapabilities**

| Elements                        | Requirement | Notes      |
|---------------------------------|-------------|------------|
| InstanceID                      | Mandatory   | <b>Key</b> |
| VirtualSystemTypesSupported[ ]  | Optional    | See 7.4.2. |
| SynchronousMethodsSupported[ ]  | Optional    | See 7.4.3. |
| AsynchronousMethodsSupported[ ] | Optional    | See 7.4.4. |
| IndicationsSupported[ ]         | Optional    | See 7.4.5. |

2398 **10.21 CIM\_VirtualSystemManagementService**

2399 The implementation of the CIM\_VirtualSystemManagementService class is conditional.

2400 Condition: Any of the following is implemented:

- 2401 • Virtual system definition and destruction (see 7.4.6.1)
- 2402 • Virtual resource addition and removal (see 7.4.6.2)
- 2403 • Virtual system and resource modification (see 7.4.6.3)

2404 If the CIM\_VirtualSystemManagementService class is implemented, the provisions in this subclause  
2405 apply.

2406 An implementation shall use an instance of the CIM\_VirtualSystemManagementService class to  
2407 represent the virtual system management service provided by one host system.

2408 Table 41 contains requirements for elements of this class.

2409 **Table 41 – Class: CIM\_VirtualSystemManagementService**

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

2410 **10.22 CIM\_VirtualSystemSettingData (Input)**

2411 The implementation of the CIM\_VirtualSystemSettingData class for input is conditional.

2412 Condition: Any of the following is implemented:

- 2413 • Virtual system definition and destruction (see 7.4.6.1)
- 2414 • Virtual resource addition and removal (see 7.4.6.2)
- 2415 • Virtual system and resource modification (see 7.4.6.3)

2416 If the CIM\_VirtualSystemSettingData class is implemented for input, the provisions in this subclause  
2417 apply.

2418 An instance of the CIM\_VirtualSystemSettingData class shall be used to represent input data for a virtual  
2419 system’s definitions and modifications.

2420 Table 42 contains requirements for elements of this class.

2421

**Table 42 – Class: CIM\_VirtualSystemSettingData (Input)**

| Elements              | Requirement | Notes                          |
|-----------------------|-------------|--------------------------------|
| InstanceID            | Mandatory   | <b>Key</b> (Input): See 7.5.1. |
| ElementName           | Optional    | See 7.5.2.                     |
| VirtualSystemIdentity | Optional    | See 7.5.3.                     |
| VirtualSystemType     | Optional    | See 7.5.4.                     |

### 2422 10.23 CIM\_VirtualSystemSettingData (Snapshot)

2423 The implementation of the CIM\_VirtualSystemSettingData class for the representation of snapshots of virtual systems is conditional.

2425 Condition: Virtual system snapshots are implemented; see 7.7.1.1.

2426 If the CIM\_VirtualSystemSettingData class is implemented for the representation of snapshots, the provisions in this subclause apply.

2428 An instance of the CIM\_VirtualSystemSettingData class shall be used to represent snapshots of virtual systems.

2430 Table 43 contains requirements for elements of this class.

2431

**Table 43 – Class: CIM\_VirtualSystemSettingData (Snapshot)**

| Elements                             | Requirement | Notes  |
|--------------------------------------|-------------|--|
| InstanceID                           | Mandatory   | <b>Key</b>   |
| 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.                   |



| Elements   | Requirement | Notes                                    |
|--|-------------|--|
| 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. |             |  |

2432 **10.24 CIM\_VirtualSystemSnapshotCapabilities**

2433 The implementation of the CIM\_VirtualSystemSnapshotCapabilities class is optional.

2434 If the CIM\_VirtualSystemSnapshotCapabilities class is implemented, the provisions in this subclause  
2435 apply.

2436 The implementation of the optional CIM\_VirtualSystemSnapshotCapabilities class is specified only if vir-  
2437 tual system snapshots are implemented; see 7.7.1.1.

2438 An instance of the CIM\_VirtualSystemSnapshotCapabilities class may be used to represent the current  
2439 applicability of snapshot-related services to one virtual system.

2440 Table 44 contains requirements for elements of this class.

2441 **Table 44 – Class: CIM\_VirtualSystemSnapshotCapabilities**

| Elements                      | Requirement | Notes        |
|-------------------------------|-------------|--------------|
| InstanceID                    | Mandatory   | <b>Key</b>   |
| SnapshotTypesEnabled[ ]       | Mandatory   | See 7.7.5.1. |
| GuestOSNotificationEnabled[ ] | Optional    | See 7.7.5.2. |

2442 **10.25 CIM\_VirtualSystemSnapshotService**

2443 The implementation of the CIM\_VirtualSystemSnapshotService class is optional.

2444 If the CIM\_VirtualSystemSnapshotService class is implemented, the provisions in this subclause apply.

2445 If the CIM\_VirtualSystemSnapshotService class is implemented, this indicates the presence of the sup-  
2446 port of virtual system snapshots (see 7.7.1.1).

2447 An instance of the CIM\_VirtualSystemSnapshotService class shall be used to represent the virtual system  
2448 snapshot service available at a host system.

2449 Table 45 contains requirements for elements of this class.

2450 **Table 45 – Class: CIM\_VirtualSystemSnapshotService**

| Elements                | Requirement | Notes      |
|-------------------------|-------------|------------|
| CreationClassName       | Mandatory   | <b>Key</b> |
| Name                    | Mandatory   | <b>Key</b> |
| SystemCreationClassName | Mandatory   | <b>Key</b> |
| SystemName              | Mandatory   | <b>Key</b> |

| Elements           | Requirement | Notes      |
|--------------------|-------------|------------|
| CreateSnapshot( )  | Conditional | See 8.3.1. |
| DestroySnapshot( ) | Conditional | See 8.3.2. |
| ApplySnapshot( )   | Conditional | See 8.3.3. |

2451 **10.26 CIM\_VirtualSystemSnapshotServiceCapabilities**

2452 The implementation of the CIM\_VirtualSystemSnapshotServiceCapabilities class is conditional.

2453 Condition: Virtual system snapshots are implemented; see 7.7.1.1.

2454 If the CIM\_VirtualSystemSnapshotServiceCapabilities class is implemented, the provisions in this  
2455 subclause apply.

2456 An instance of the CIM\_VirtualSystemSnapshotServiceCapabilities class shall be used to represent the  
2457 capabilities of a virtual system snapshot service.

2458 Table 46 contains requirements for elements of this class.

2459 **Table 46 – Class: CIM\_VirtualSystemSnapshotServiceCapabilities**

| Elements                        | Requirement | Notes        |
|---------------------------------|-------------|--------------|
| InstanceID                      | Mandatory   | <b>Key</b>   |
| SynchronousMethodsSupported[ ]  | Conditional | See 7.7.1.2. |
| AsynchronousMethodsSupported[ ] | Conditional | See 7.7.1.2. |
| SnapshotTypesSupported[ ]       | Mandatory   | See 7.7.1.2. |

2460

**ANNEX A  
(Informative)****Change Log**2461  
2462  
2463  
2464

| Version | Date       | Description                      |
|---------|------------|----------------------------------|
| 1.0.0a  | 2007-08-03 | Released as preliminary standard |
| 1.0.0   | 2010-04-22 | Released as DMTF Standard        |
|         |            |                                  |

2465  
2466