



1
2
3
4

Document Number: DSP0813

Date: 2009-06-04

Version: 1.0.0

5 **Boot Control Profile SM CLP Command Mapping**
6 **Specification**

7 **Document Type: Specification**
8 **Document Status: DMTF Standard**
9 **Document Language: E**

10

11 Copyright notice

12 Copyright © 2006, 2009 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

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

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27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

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

33

34

CONTENTS

35 Foreword 5

36 Introduction 6

37 1 Scope 7

38 2 Normative References..... 7

39 2.1 Approved References 7

40 2.2 Other References..... 7

41 3 Terms and Definitions..... 7

42 4 Symbols and Abbreviated Terms..... 8

43 5 Recipes..... 9

44 6 Mappings..... 9

45 6.1 CIM_BootService..... 9

46 6.2 CIM_BootServiceCapabilities 12

47 6.3 CIM_BootConfigSetting 14

48 6.4 CIM_BootSettingData 28

49 6.5 CIM_BootSourceSetting 30

50 6.6 CIM_ConcreteComponent 32

51 6.7 CIM_ConcreteDependency 36

52 6.8 CIM_ElementCapabilities 38

53 6.9 CIM_ElementSettingData 41

54 6.10 CIM_HostedService 47

55 6.11 CIM_LogicalIdentity 50

56 6.12 CIM_OrderedComponent 52

57 6.13 CIM_ServiceAffectsElement 55

58 ANNEX A (informative) Change Log 58

59

60 Tables

61 Table 1 – Command Verb Requirements for CIM_BootService 10

62 Table 2 – Command Verb Requirements for CIM_BootServiceCapabilities 13

63 Table 3 – Command Verb Requirements for CIM_BootConfigSetting 15

64 Table 4 – Command Verb Requirements for CIM_BootSettingData 28

65 Table 5 – Command Verb Requirements for CIM_BootSourceSetting 30

66 Table 6 – Command Verb Requirements for CIM_ConcreteComponent 33

67 Table 7 – Command Verb Requirements for CIM_ConcreteDependency 36

68 Table 8 – Command Verb Requirements for CIM_ElementCapabilities 38

69 Table 9 – Command Verb Requirements for CIM_ElementSettingData 41

70 Table 10 – Command Verb Requirements for CIM_HostedService 48

71 Table 11 – Command Verb Requirements for CIM_LogicalIdentity 50

72 Table 12 – Command Verb Requirements for CIM_OrderedComponent 53

73 Table 13 – Command Verb Requirements for CIM_ServiceAffectsElement 55

74

76

Foreword

77 The *Boot Control Profile SM CLP Command Mapping Specification* (DSP0813) was prepared by the
78 Server Management Working Group.

79 **Conventions**

80 The pseudo code conventions utilized in this document are the Recipe Conventions as defined in the
81 SNIA [SMI-S 1.1.0](#), section 7.6.

82 **Acknowledgements**

83 The authors wish to acknowledge the following participants from the DTMF Server Management Working
84 Group:

- 85 • John Leung – Intel
- 86 • Aaron Merkin – IBM
- 87 • Christina Shaw – HP
- 88 • Enoch Suen – Dell
- 89 • Jon Hass – Dell
- 90 • Jeff Hilland – HP
- 91 • Khachatur Papanyan – Dell
- 92 • Arvind Kumar – Intel
- 93 • Perry Vincent – Intel

94

95

Introduction

96 This document defines the SM CLP mapping for CIM elements described in the [Boot Control Profile](#). The
97 information in this specification, combined with the [SM CLP-to-CIM Common Mapping Specification 1.0](#),
98 is intended to be sufficient to implement SM CLP commands relevant to the classes, properties, and
99 methods described in the [Boot Control Profile](#) using CIM operations.

100 The target audience for this specification is implementers of the SM CLP support for the [Boot Control](#)
101 [Profile](#).

102
103

Boot Control Profile SM CLP Command Mapping Specification

104 1 Scope

105 This specification contains the requirements for an implementation of the SM CLP to provide access to,
106 and implement the behaviors of, the [Boot Control Profile](#).

107 2 Normative References

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

111 2.1 Approved References

112 DMTF DSP1012, *Boot Control Profile 1.0*,
113 http://www.dmtf.org/standards/published_documents/DSP1012_1.0.pdf

114 DMTF DSP0216, *SM CLP-to-CIM Common Mapping Specification 1.0*,
115 http://www.dmtf.org/standards/published_documents/DSP0216_1.0.pdf

116 SNIA, *Storage Management Initiative Specification (SMI-S) 1.1.0*,
117 http://www.snia.org/tech_activities/standards/curr_standards/smi

118 2.2 Other References

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

121 3 Terms and Definitions

122 For the purposes of this document, the following terms and definitions apply.

123 3.1

124 **can**

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

126 3.2

127 **cannot**

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

129 3.3

130 **conditional**

131 indicates requirements to be followed strictly in order to conform to the document when the specified
132 conditions are met

- 133 **3.4**
134 **mandatory**
135 indicates requirements to be followed strictly in order to conform to the document and from which no
136 deviation is permitted
- 137 **3.5**
138 **may**
139 indicates a course of action permissible within the limits of the document
- 140 **3.6**
141 **need not**
142 indicates a course of action permissible within the limits of the document
- 143 **3.7**
144 **optional**
145 indicates a course of action permissible within the limits of the document
- 146 **3.8**
147 **shall**
148 indicates requirements to be followed strictly in order to conform to the document and from which no
149 deviation is permitted
- 150 **3.9**
151 **shall not**
152 indicates requirements to be followed strictly in order to conform to the document and from which no
153 deviation is permitted
- 154 **3.10**
155 **should**
156 indicates that among several possibilities, one is recommended as particularly suitable, without
157 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 158 **3.11**
159 **should not**
160 indicates that a certain possibility or course of action is deprecated but not prohibited

161 **4 Symbols and Abbreviated Terms**

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

- 163 **4.1**
164 **CIM**
165 Common Information Model
- 166 **4.2**
167 **CLP**
168 Command Line Protocol
- 169 **4.3**
170 **DMTF**
171 Distributed Management Task Force

172 **4.4**
173 **IETF**
174 Internet Engineering Task Force

175 **4.5**
176 **SM**
177 Server Management

178 **4.6**
179 **SMI-S**
180 Storage Management Initiative Specification

181 **4.7**
182 **SNIA**
183 Storage Networking Industry Association

184 **4.8**
185 **UFsT**
186 User Friendly selection Tag

187 **5 Recipes**

188 The following is a list of the common recipes used by the mappings in this specification. For a definition of
189 each recipe, see *SM CLP-to-CIM Common Mapping Specification 1.0* ([DSP0216](#)).

- 190 • smOpDeleteInstance()
- 191 • smReset()
- 192 • smShowInstance()
- 193 • smShowInstances()
- 194 • smSetInstance()
- 195 • smShowAssociationInstances()

196 This mapping does not define any recipes for local reuse.

197 **6 Mappings**

198 The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in
199 the [Boot Control Profile](#). Requirements specified here related to support for a CLP verb for a particular
200 class are solely within the context of this profile.

201 **6.1 CIM_BootService**

202 The `cd` and `help` verbs shall be supported as described in [DSDP0216](#).

203 Table 1 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
204 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
205 target. Table 1 is for informational purposes only; in case of a conflict between Table 1 and requirements
206 detailed in the following sections, the text detailed in the following sections supersedes the information in
207 Table 1.

208

Table 1 – Command Verb Requirements for CIM_BootService

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	May	See 6.1.2.
show	Shall	See 6.1.3.
start	Not supported	
stop	Not supported	

209 No mappings are defined for the following verbs for the specified target: create, delete, dump, exit,
 210 load, reset, start, and stop.

211 **6.1.1 Ordering of Results**

212 When results are returned for multiple instances of CIM_BootService, implementations shall utilize the
 213 following algorithm to produce the natural (that is, default) ordering:

- 214 • Results for CIM_BootService are unordered; therefore, no algorithm is defined.

215 **6.1.2 Set**

216 The set verb is used to set properties on an instance of CIM_BootService. Implementations may support
 217 the use of the set verb with CIM_BootService.

218 **6.1.2.1 General Usage of Set for a Single Property**

219 This command form corresponds to the general usage of the set verb to modify a single property of a
 220 target instance.

221 The requirements for supporting modification of a property using this command form shall be equivalent
 222 to the requirement for supporting modification of the property using the ModifyInstance operation as
 223 defined in the [Boot Control Profile](#).

224 **6.1.2.1.1 Command Form**

225 `set <CIM_BootService single instance> <propertyname>=<propertyvalue>`

226 **6.1.2.1.2 CIM Requirements**

227 See CIM_BootService in the “CIM Elements” section of the [Boot Control Profile](#) for the list of mandatory
 228 properties.

229 6.1.2.1.3 Behavior Requirements

230 6.1.2.1.3.1 Pseudo Code

```

231 $instance=<CIM_BootService single instance>
232 #propertyName[] = <propertname>
233 #propertyValues[] = <propertyvalue>
234 &smSetInstance ( $instance, #propertyName, #propertyValues );
235 &smEnd;
```

236 6.1.2.2 General Usage of Set for Multiple Properties

237 This command form corresponds to the general usage of the `set` verb to modify a multiple properties of a
 238 target instance where there isn't an explicit relationship between the properties.

239 The requirements for supporting modification of a property using this command form shall be equivalent
 240 to the requirement for supporting modification of the property using the `ModifyInstance` operation as
 241 defined in the [Boot Control Profile](#).

242 6.1.2.2.1 Command Form

```

243 set <CIM_BootService single instance> <propertyname1>=<propertyvalue1>
244 <propertynameN>=<propertyvalueN>
```

245 6.1.2.2.2 CIM Requirements

246 See `CIM_BootService` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of mandatory
 247 properties.

248 6.1.2.2.3 Behavior Requirements

249 6.1.2.2.3.1 Preconditions

250 `$instance` represents the instance of `CIM_BootService`.

251 6.1.2.2.3.2 Pseudo Code

```

252 for #i < n
253 {
254   #propertyName[#i] = <propertname#i>
255   #propertyValues[#i] = <propertyvalue#i>
256 }
257 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
258 &smEnd;
```

259 6.1.3 Show

260 The `show` verb is used to display information about instances of `CIM_BootService`. Implementations shall
 261 support the use of the `show` verb with `CIM_BootService`.

262 6.1.3.1 Show a Single Instance

263 This command form is used to display information about a single instance of `CIM_BootService`.

264 6.1.3.1.1 Command Form

```

265 show <CIM_BootService single instance>
```

266 6.1.3.1.2 CIM Requirements

267 See CIM_BootService in the “CIM Elements” section of the [Boot Control Profile](#) for the list of mandatory
268 properties.

269 6.1.3.1.3 Behavior Requirements

270 6.1.3.1.3.1 Preconditions

271 \$instance represents the instance of CIM_BootService.

272 #all is true, if the “-all” option was specified with the command; otherwise, #all is false.

273 #propertylist[] is an array of mandatory non-key property names.

274 6.1.3.1.3.2 Pseudo Code

```
275 if (false != #all) { #propertylist[] = NULL; }  
276 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );  
277 &smEnd;
```

278 6.1.3.2 Show Multiple Instances

279 This command form is used to display information about multiple instances of CIM_BootService.

280 6.1.3.2.1 Command Form

```
281 show <CIM_BootService multiple instances>
```

282 6.1.3.2.2 CIM Requirements

283 See CIM_BootService in the “CIM Elements” section of the [Boot Control Profile](#) for the list of mandatory
284 properties.

285 6.1.3.2.3 Behavior Requirements

286 6.1.3.2.3.1 Preconditions

287 \$containerInstance represents the instance of CIM_ComputerSystem to which the instances of
288 CIM_BootService are scoped. The CIM_BootService is associated to CIM_ComputerSystem via a
289 CIM_HostedService association.

290 #all is true, if the “-all” option was specified with the command; otherwise, #all is false.

291 #propertylist[] is an array of mandatory non-key property names.

292 6.1.3.2.3.2 Pseudo Code

```
293 if (false != #all) { #propertylist[] = NULL; }  
294 &smShowInstances ( "CIM_BootService", "CIM_HostedService",  
295 $containerInstance.getObjectPath(), #propertylist[] );  
296 &smEnd;
```

297 6.2 CIM_BootServiceCapabilities

298 The cd and help verbs shall be supported as described in [DSP0216](#).

299 Table 2 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
 300 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
 301 target. Table 2 is for informational purposes only; in case of a conflict between Table 2 and requirements
 302 detailed in the following sections, the text detailed in the following sections supersedes the information in
 303 Table 2.

304 **Table 2 – Command Verb Requirements for CIM_BootServiceCapabilities**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.2.2.
start	Not supported	
stop	Not supported	

305 No mappings are defined for the following verbs for the specified target: create, delete, dump, exit,
 306 load, reset, set, start, and stop.

307 **6.2.1 Ordering of Results**

308 When results are returned for multiple instances of CIM_BootServiceCapabilities, implementations shall
 309 utilize the following algorithm to produce the natural (that is, default) ordering:

- 310 • Results for CIM_BootServiceCapabilities are unordered; therefore, no algorithm is defined.

311 **6.2.2 Show**

312 The `show` verb is used to display information about instances of CIM_BootServiceCapabilities.
 313 Implementations shall support the use of the `show` verb with CIM_BootServiceCapabilities.

314 **6.2.2.1 Show a Single Instance**

315 This command form is used to display the information about a single instance of
 316 CIM_BootServiceCapabilities.

317 **6.2.2.1.1 Command Form**

318 `show <CIM_BootServiceCapabilities single instance>`

319 **6.2.2.1.2 CIM Requirements**

320 See CIM_BootServiceCapabilities in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
 321 mandatory properties.

322 6.2.2.1.3 Behavior Requirements

323 6.2.2.1.3.1 Preconditions

324 \$instance represents the instance of CIM_BootServiceCapabilities.

325 #all is true, if the “-all” option was specified with the command; otherwise, #all is false.

326 #propertylist[] is an array of mandatory non-key property names.

327 6.2.2.1.3.2 Pseudo Code

```
328 if (false != #all) { #propertylist[] = NULL; }
329 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
330 &smEnd;
```

331 6.2.2.2 Show Multiple Instances

332 This command form is used to display the information about multiple instances of
333 CIM_BootServiceCapabilities. This command form corresponds to UFT-based selection within a scoping
334 system.

335 6.2.2.2.1 Command Form

```
336 show <CIM_BootServiceCapabilities multiple instances>
```

337 6.2.2.2.2 CIM Requirements

338 See CIM_BootServiceCapabilities in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
339 mandatory properties.

340 6.2.2.2.3 Behavior Requirements

341 6.2.2.2.3.1 Preconditions

342 \$containerInstance represents the instance of CIM_ConcreteCollection with ElementName property
343 that contains “Capabilities” and is associated to the targeted instances of CIM_BootServiceCapabilities
344 through the CIM_MemberOfCollection association.

345 #all is true, if the “-all” option was specified with the command; otherwise, #all is false.

346 #propertylist[] is an array of mandatory non-key property names.

347 6.2.2.2.3.2 Pseudo Code

```
348 if (false != #all) { #propertylist[] = NULL; }
349 &smShowInstances ( "CIM_BootServiceCapabilities", "CIM_MemberOfCollection",
350   $containerInstance.getObjectPath(), #propertylist[] );
351 &smEnd;
```

352 6.3 CIM_BootConfigSetting

353 The cd and help verbs shall be supported as described in [DSP0216](#).

354 Table 3 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
355 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
356 target. Table 3 is for informational purposes only; in case of a conflict between Table 3 and requirements
357 detailed in the following sections, the text detailed in the following sections supersedes the information in
358 Table 3.

359

Table 3 – Command Verb Requirements for CIM_BootConfigSetting

Command Verb	Requirement	Comments
create	May	See 6.3.2.
delete	May	See 6.3.3.
dump	Not supported	
load	Not supported	
reset	Not supported	
set	May	See 6.3.4.
show	Shall	See 6.3.5.
start	Not supported	
stop	Not supported	

360 No mappings are defined for the following verbs for the specified target: dump, exit, load, reset,
 361 start, and stop.

362 **6.3.1 Ordering of Results**

363 When results are returned for multiple instances of CIM_BootConfigSetting, implementations shall utilize
 364 the following algorithm to produce the natural (that is, default) ordering:

- 365 • Results for CIM_BootConfigSetting are unordered; therefore, no algorithm is defined.

366 **6.3.2 Create**

367 The `create` verb is used to create an instance of CIM_BootConfigSetting and associated instances
 368 using a template CIM_BootConfigSetting instance. Implementations may support the use of the `create`
 369 verb with CIM_BootConfigSetting.

370 The template CIM_BootConfigSetting instance can be specified in the command. Otherwise, the template
 371 CIM_BootConfigSetting instance shall be the CIM_BootConfigSetting instance whose
 372 CIM_ElementSettingData.IsCurrent property has the value of 1 (IsDefault).

373 **6.3.2.1 Create a Single Instance from a Specified Boot Configuration**

374 This command form is used to create an instance of CIM_BootConfigSetting using an existing
 375 CIM_BootConfigSetting as the template.

376 **6.3.2.1.1 Command Form**

377 `create -source <BootConfigTemplatePath> <CIM_BootConfigSetting single instance>`

378 **6.3.2.1.2 CIM Requirements**

379 See CIM_BootConfigSetting in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
 380 mandatory properties.

381 **6.3.2.1.3 Behavior Requirements**

382 **6.3.2.1.3.1 Preconditions**

383 `$instanceTemplatePath` represents a path to an instance of CIM_BootConfigSetting to use as the
 384 creation template. `$instanceTemplatePath` is translated from the `<BootConfigTemplatePath>`.

385 \$computerInstance represents the instance of CIM_ComputerSystem which is associated to Setting
 386 Collection via a CIM_OwningCollectionElement association. The Setting Collection is the instance of
 387 CIM_ConcreteCollection whose ElementName property is set to "Settings" which can be translated from
 388 the <CIM_BootConfigSetting single instance> path.

389 #propertylist[] is an array of mandatory non-key property names.

390 6.3.2.1.3.2 Pseudo Code

```

391 //Step 1: Verify Template Boot Configuration exists
392 #Error = &smOpGetInstance ($instanceTemplatePath->, NULL, $instanceTemplate);
393 if (0 != Error.code)
394     {
395         &smProcessOpError (#Error);
396     }
397 // Step 2: Try to find the CIM_ComputerSystem
398 #Error = smOpAssociators($instanceTemplatePath.getObjectPath(),
399     "CIM_ElementSettingData", "CIM_ComputerSystem", "ManagedElement", "SettingData",
400     NULL, $System->[]);
401 if (0 != Error.code)
402     {
403         &smProcessOpError (#Error);
404     }
405 //no system associated
406 if (null == $System[0])
407     {
408         $OperationError = smNewInstance("CIM_Error");
409         //CIM_ERR_FAILED
410         $OperationError.CIMStatusCode = 1;
411         //Software Error
412         $OperationError.ErrorType = 4;
413         //Unknown
414         $OperationError.PerceivedSeverity = 0;
415         $OperationError.OwningEntity = DMTF:SMCLP;
416         $OperationError.MessageID = 0x00000001;
417         $OperationError.Message = "Operation is not supported";
418         &smAddError($job, $OperationError);
419         &smMakeCommandStatus($job);
420         &smEnd;
421     }
422 // Step 2: Try to find the CIM_BootService
423 #Error = smOpAssociators($System[0].getObjectPath(), "CIM_ServiceAffectsElement",
424     "CIM_BootService", "AffectingElement", "AffectedElement", NULL, Services->[]);
425 if (0 != Error.code)
426     {
427         &smProcessOpError (#Error);
428     }
429 //no service associated
430 if (null == Services[0])
431     {
432         $OperationError = smNewInstance("CIM_Error");
  
```



```

433 //CIM_ERR_FAILED
434 $OperationError.CIMStatusCode = 1;
435 //Software Error
436 $OperationError.ErrorType = 4;
437 //Unknown
438 $OperationError.PerceivedSeverity = 0;
439 $OperationError.OwningEntity = DMTF:SMCLP;
440 $OperationError.MessageID = 0x00000001;
441 $OperationError.Message = "Operation is not supported";
442 &smAddError($job, $OperationError);
443 &smMakeCommandStatus($job);
444 &smEnd;
445 }
446 //Take the first instance we find
447 $Service-> = $Services[0].getObjectPath();
448 //Step 2: Build parameter lists for method invocation
449 %InArguments[] =
450 {
451     newArgument("ScopingComputerSystem", $computerInstance)
452     newArgument("StartBootConfig", $instanceTemplatePath.getObjectPath())
453 };
454 %OutArguments[] =
455 {
456     newArgument("Job", instanceConcreteJob.getObjectPath())
457     newArgument("NewBootConfig", $instanceBootConfigSetting)
458 };
459 //Step 3: Invoke method
460 #Error = smOpInvokeMethod ($Service->, "CreateBootConfig", %InArguments[],
461     %OutArguments[], returnStatus);
462 //Step 4: Process return code to CLP Command Status
463 if (0 != #Error.code) {
464 //method invocation failed
465     if ( (null != #Error.$error) && (null != #Error.$error[0]) ) {
466         //if the method invocation contains an embedded error
467         //use it for the Error for the overall job
468         &smAddError($job, #Error.$error[0]);
469         &smMakeCommandStatus($job);
470         &smEnd;
471     } else if ( 17 == #Error.code ) {
472         //17 - CIM_ERR_METHOD_NOT_FOUND
473         // The specified extrinsic method does not exist.
474         $OperationError = smNewInstance("CIM_Error");
475         // CIM_ERR_METHOD_NOT_FOUND
476         $OperationError.CIMStatusCode = 17;
477         //Software Error
478         $OperationError.ErrorType = 10;
479         //Unknown
480         $OperationError.PerceivedSeverity = 0;
481         $OperationError.OwningEntity = DMTF:SMCLP;

```

```

482     $OperationError.Message = "Operation is not supported."
483     &smAddError($job, $OperationError);
484     &smMakeCommandStatus($job);
485     &smEnd;
486 } else {
487     //operation failed, but no detailed error instance, need to make one
488     //up make an Error instance and associate with job for Operation
489     $OperationError = smNewInstance("CIM_Error");
490     //CIM_ERR_FAILED
491     $OperationError.CIMStatusCode = 1;
492     //Software Error
493     $OperationError.ErrorType = 4;
494     //Unknown
495     $OperationError.PerceivedSeverity = 0;
496     $OperationError.OwningEntity = DMTF:SMCLP;
497     $OperationError.MessageID = 0x00000009;
498     $OperationError.Message = "An internal software error has occurred.";
499     &smAddError($job, $OperationError);
500     &smMakeCommandStatus($job);
501     &smEnd;
502 }
503 } else if (0 == #returnStatus) {
504     // Method completed successfully, show new boot configuration
505     &smShowInstance($instance.GetObjectPath(), #propertylist[] );
506     &smCommandCompleted($job);
507     &smEnd;
508 } else if (4096 == #returnStatus) {
509     //job spawned, need to watch for it to finish
510     //while the jobstate is 4 ("Running")
511     while (4 == $instanceConcreteJob.JobState){<busy wait>}
512     //when job finishes, invoke GetError()
513     if (2 != $job.OperationalStatus) {
514         %InArguments[] = { }
515         %OutArguments[] = {newArgument("Job", $instanceConcreteJob.GetObjectPath())}
516         #Error = smOpInvokeMethod($job, "GetError", %InArguments, %OutArguments,
517             #returncode);
518
519         //Method invocation failed, internal processing error
520         if ( (0 != #Error.code) || (0 != #returncode) ) {
521             //make an Error instance and associate with job for Operation
522             $OperationError = smNewInstance("CIM_Error");
523             //CIM_ERR_FAILED
524             $OperationError.CIMStatusCode = 1;
525             //Software Error
526             $OperationError.ErrorType = 4;
527             //Unknown
528             $OperationError.PerceivedSeverity = 0;
529             $OperationError.OwningEntity = DMTF:SMCLP;
530             $OperationError.MessageID = 0x00000009;

```

```

531     $OperationError.Message = "An internal software error has occurred.";
532     &smAddError($job, $OperationError);
533     &smMakeCommandStatus($job);
534     &smEnd;
535   } else {
536     //make command status
537     $joberror = %OutArguments["Error"];
538     &smMakeCommandExecutionFailed($job, {$joberror});
539   } //end if have CIM_Error from GetError()
540 }
541 } else {
542   //unspecified return code, generic failure.
543   $OperationError = smNewInstance("CIM_Error");
544   //CIM_ERR_FAILED
545   $OperationError.CIMStatusCode = 1;
546   //Other
547   $OperationError.ErrorType = 1;
548   //Low
549   $OperationError.PerceivedSeverity = 2;
550   $OperationError.OwningEntity = DMTF:SMCLP;
551   $OperationError.MessageID = 0x00000002;
552   $OperationError.Message = "Failed. No further information is available.";
553   &smAddError($job, $OperationError);
554   &smMakeCommandStatus($job);
555   &smEnd;
556 }

```

557 6.3.2.2 Create a Single Instance from the Default Boot Configuration

558 This command form is used to create an instance of CIM_BootConfigSetting using an existing
 559 CIM_BootConfigSetting with the role of "Is Default".

560 6.3.2.2.1 Command Form

```
561 create <CIM_BootConfigSetting single instance>
```

562 6.3.2.2.2 CIM Requirements

563 See CIM_BootConfigSetting in the "CIM Elements" section of the [Boot Control Profile](#) for the list of
 564 mandatory properties.

565 6.3.2.2.3 Behavior Requirements

566 6.3.2.2.3.1 Preconditions

567 \$computerInstance represents the instance of CIM_ComputerSystem which is associated to Setting
 568 Collection via a CIM_OwningCollectionElement association. The Setting Collection is the instance of
 569 CIM_ConcreteCollection whose ElementName property is set to "Settings" which can be translated from
 570 <CIM_BootConfigSetting single instance> path.

571 #propertylist[] is an array of mandatory non-key property names.

572 **6.3.2.2.4 Pseudo Code**

```

573 // Step 1: Try to find the CIM_BootService
574 #Error = smOpAssociators($computerInstance.getObjectPath(),
575     "CIM_ServiceAffectsElement", "CIM_BootService", "AffectingElement",
576     "AffectedElement", NULL, Services->[]);
577 if (0 != Error.code)
578     {
579     &smProcessOpError (#Error);
580     }
581 //no service associated
582 if (null == Services[0])
583     {
584     $OperationError = smNewInstance("CIM_Error");
585     //CIM_ERR_FAILED
586     $OperationError.CIMStatusCode = 1;
587     //Software Error
588     $OperationError.ErrorType = 4;
589     //Unknown
590     $OperationError.PerceivedSeverity = 0;
591     $OperationError.OwningEntity = DMTF:SMCLP;
592     $OperationError.MessageID = 0x00000001;
593     $OperationError.Message = "Operation is not supported.";
594     &smAddError($job, $OperationError);
595     &smMakeCommandStatus($job);
596     &smEnd;
597     }
598 //Take the first instance we find
599 $Service-> = $Services[0].getObjectPath();
600 //Step 2: Build parameter lists for method invocation
601 %InArguments[] = {
602     newArgument("ScopingComputerSystem", $computerInstance)
603     newArgument("StartBootConfig", NULL)
604 };
605 %OutArguments[] = {
606     newArgument("Job", instanceConcreteJob.getObjectPath())
607     newArgument("NewBootConfig", $instanceBootConfigSetting)
608 };
609 // Step 3: Invoke method
610 #Error = smOpInvokeMethod ($Service->, "CreateBootConfig", %InArguments[],
611     %OutArguments[], returnStatus);
612 // Step 4: Process return code to CLP Command Status
613 if (0 != #Error.code) {
614     //method invocation failed
615     if ( (null != #Error.$error) && (null != #Error.$error[0]) )    {
616         //if the method invocation contains an embedded error
617         //use it for the Error for the overall job
618         &smAddError($job, #Error.$error[0]);
619         &smMakeCommandStatus($job);
620         &smEnd;

```

```

621 } else if ( 17 == #Error.code ) {
622     //17 - CIM_ERR_METHOD_NOT_FOUND
623     // The specified extrinsic method does not exist.
624     $OperationError = smNewInstance("CIM_Error");
625     // CIM_ERR_METHOD_NOT_FOUND
626     $OperationError.CIMStatusCode = 17;
627     //Software Error
628     $OperationError.ErrorType = 10;
629     //Unknown
630     $OperationError.PerceivedSeverity = 0;
631     $OperationError.OwningEntity = DMTF:SMCLP;
632     $OperationError.MessageID = 0x00000001;
633     $OperationError.Message = "Operation is not supported."
634     &smAddError($job, $OperationError);
635     &smMakeCommandStatus($job);
636     &smEnd;
637 } else {
638     //operation failed, but no detailed error instance, need to make one
639     //up make an Error instance and associate with job for Operation
640     $OperationError = smNewInstance("CIM_Error");
641     //CIM_ERR_FAILED
642     $OperationError.CIMStatusCode = 1;
643     //Software Error
644     $OperationError.ErrorType = 4;
645     //Unknown
646     $OperationError.PerceivedSeverity = 0;
647     $OperationError.OwningEntity = DMTF:SMCLP;
648     $OperationError.MessageID = 0x00000009;
649     $OperationError.Message = "An internal software error has occurred.";
650     &smAddError($job, $OperationError);
651     &smMakeCommandStatus($job);
652     &smEnd;
653 }
654 } else if ( 0 == #returnStatus) {
655     // Method completed successfully, show new boot configuration
656     &smDisplayInstance ($instanceBootConfigSetting);
657     &smCommandCompleted($job);
658     &smEnd;
659 } else if ( 4096 == #returnStatus) {
660     //job spawned, need to watch for it to finish
661     //while the jobstate is 4 ("Running")
662     while ( 4 == $instanceConcreteJob.JobState){<busy wait>}
663     //when job finishes, invoke GetError()
664     if ( 2 != $job.OperationalStatus) {
665         %InArguments[] = { }
666         %OutArguments[] = {newArgument("Job", $instanceConcreteJob.getObjectPath())}
667         #Error = smOpInvokeMethod($job, "GetError", %InArguments, %OutArguments,
668             #returncode);
669         //Method invocation failed, internal processing error

```

```

670     if ( (0 != #Error.code) || (0 != #returncode) ) {
671         //make an Error instance and associate with job for Operation
672         $OperationError = smNewInstance("CIM_Error");
673         //CIM_ERR_FAILED
674         $OperationError.CIMStatusCode = 1;
675         //Software Error
676         $OperationError.ErrorType = 4;
677         //Unknown
678         $OperationError.PerceivedSeverity = 0;
679         $OperationError.OwningEntity = DMTF:SMCLP;
680         $OperationError.MessageID = 0x00000009;
681         $OperationError.Message = "An internal software error has occurred.";
682         &smAddError($job, $OperationError);
683         &smMakeCommandStatus($job);
684         &smEnd;
685     } else {
686         //make command status
687         $joberror = %OutArguments["Error"];
688         &smMakeCommandExecutionFailed($job, {$joberror});
689     } //end if have CIM_Error from GetError()
690 } //embedded job not OK
691 } else {
692     //unspecified return code, generic failure.
693     $OperationError = smNewInstance("CIM_Error");
694     //CIM_ERR_FAILED
695     $OperationError.CIMStatusCode = 1;
696     //Other
697     $OperationError.ErrorType = 1;
698     //Low
699     $OperationError.PerceivedSeverity = 2;
700     $OperationError.OwningEntity = DMTF:SMCLP;
701     $OperationError.MessageID = 0x00000002;
702     $OperationError.Message = "Failed. No further information is available.";
703     &smAddError($job, $OperationError);
704     &smMakeCommandStatus($job);
705     &smEnd;
706 }

```

6.3.3 Delete

The delete verb is used to delete instances CIM_BootConfigSetting.

Implementations may support the use of the delete verb with CIM_BootConfigSetting.

6.3.3.1 Delete a Single Instance

This command form is used to delete a single instance of CIM_BootConfigSetting.

6.3.3.1.1 Command Form

```
delete <CIM_BootConfigSetting single instance>
```

714 6.3.3.1.2 CIM Requirements

715 See CIM_BootConfigSetting in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
716 mandatory properties.

717 6.3.3.1.3 Behavior Requirements

718 6.3.3.1.3.1 Preconditions

719 \$instance represents the instance of CIM_BootConfigSetting.

720 6.3.3.1.3.2 Pseudo Code

```
721 #Error = &smOpDeleteInstance ($instance);
722 if (0 != Error.code)
723     {
724         &smProcessOpError (#Error);
725     }
726 &smEnd;
```

727 6.3.4 Set

728 This section describes how to implement the `set` verb when it is applied to an instance
729 CIM_BootConfigSetting. Implementations may support the use of the `set` verb with
730 CIM_BootConfigSetting.

731 6.3.4.1 Set the BootOrder Referenced Property

732 This command form is used to change the boot order within a Boot Configuration. The *bootorder*
733 referenced property is the mechanism of providing the boot sequence to the command. On the command
734 line, the *bootorder* referenced property is set to a list of CIM_BootSourceSetting.ElementName
735 properties. The order of the list is interpreted as the boot sequence. Each item on the list is verified to
736 match a CIM_BootSourceSetting associated to the target CIM_BootConfigSetting prior to calling the
737 ChangeBootOrder() method.

738 6.3.4.1.1 Command Form

```
739 set <CIM_BootConfigSetting single instance> bootorder=<BootSource1>,...,<BootSourceN>
```

740 6.3.4.1.2 CIM Requirements

741 See CIM_BootConfigSetting in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
742 mandatory properties.

743 6.3.4.1.3 Behavior Requirements

744 6.3.4.1.3.1 Preconditions

745 \$instance represents the instance of CIM_BootConfigSetting.

746 \$bootSources[] is an array of BootSource instances from the command line.

747 #propertylist2[] is an array of mandatory non-key property names for CIM_OrderedComponent.

748 #propertylist3[] is an array of mandatory non-key property names for CIM_BootSourceSetting.

749 6.3.4.1.3.2 Pseudo Code

```

750 // Step 1: Get the CIM_BootConfigSetting associated with each of
751 // CIM_BootSourceSetting in the boot order. Verify that
752 // CIM_BootConfigSetting for each boot source is the
753 // target CIM_BootConfigSetting.
754 // If one of the names is not found abort the command. Otherwise,
755 // save the object path in $bootOrder->[] array.
756 for #i in $bootSources[] {
757     #Error = &smOpAssociators( $bootSource[#i], "CIM_OrderedComponent",
758         "CIM_BootConfigSetting", NULL, NULL, NULL, $bootConfig[] );
759     if (0 != #Error.code) {
760         &smProcessOpError (#Error);
761         //includes &smEnd;
762     }
763     if ( $bootConfig[0] == $instance ) {
764         $bootOrder->[#i] = $bootSources[#j]->;
765         continue;
766     } else {
767         if ($instanceDefaultBootConfig == NULL) {
768             $OperationError = smNewInstance("CIM_Error");
769             $OperationError.CIMStatusCode = 1;
770             $OperationError.ErrorType = 4;
771             $OperationError.PerceivedSeverity = 0;
772             $OperationError.OwningEntity = DMFT:SMCLP;
773             $OperationError.MessageID = 0x00000009;
774             $OperationError.Message = "An internal software error has occurred.";
775             &smAddError($job, $OperationError);
776             &smMakeCommandStatus($job);
777             &smEnd;
778         }
779     }
780 }
781 //Step 2: Build parameter lists for method invocation
782 %InArguments[] = {newArgument("source", $bootOrder->[])}
783 %OutArguments[] = {
784     newArgument("Job", instanceConcreteJob.getObjectPath())
785 }
786 //step 3, invoke method
787 #Error = smOpInvokeMethod ($instance->,
788     "ChangeBootOrder", %InArguments[], %OutArguments[], #returnStatus);
789 //step 4, process return code to CLP Command Status
790 if (0 != #Error.code) {
791     //method invocation failed
792     if ( (null != #Error.$error) && (null != #Error.$error[0]) )
793     {
794         //if the method invocation contains an embedded error
795         //use it for the Error for the overall job
796         &smAddError($job, #Error.$error[0]);
797         &smMakeCommandStatus($job);

```



```

798     &smEnd;
799 }
800 else if ( 17 == #Error.code )
801 {
802     //17 - CIM_ERR_METHOD_NOT_FOUND
803     // The specified extrinsic method does not exist.
804     $OperationError = smNewInstance("CIM_Error");
805     // CIM_ERR_METHOD_NOT_FOUND
806     $OperationError.CIMStatusCode = 17;
807     //Software Error
808     $OperationError.ErrorType = 10;
809     //Unknown
810     $OperationError.PerceivedSeverity = 0;
811     $OperationError.OwningEntity = DMTF:SMCLP;
812     $OperationError.MessageID = 0x00000001;
813     $OperationError.Message = "Operation is not supported."
814     &smAddError($job, $OperationError);
815     &smMakeCommandStatus($job);
816     &smEnd;
817 }
818 else
819 {
820     //operation failed, but no detailed error instance, need to make one
821     //up make an Error instance and associate with job for Operation
822     $OperationError = smNewInstance("CIM_Error");
823     //CIM_ERR_FAILED
824     $OperationError.CIMStatusCode = 1;
825     //Software Error
826     $OperationError.ErrorType = 4;
827     //Unknown
828     $OperationError.PerceivedSeverity = 0;
829     $OperationError.OwningEntity = DMTF:SMCLP;
830     $OperationError.MessageID = 0x00000009;
831     $OperationError.Message = "An internal software error has occurred.";
832     &smAddError($job, $OperationError);
833     &smMakeCommandStatus($job);
834     &smEnd;
835 }
836 }//if CIM op failed
837 else if (0 == #returnStatus)
838 {
839     //completed successfully
840     // Show boot sources in order
841     &smOpReferences($instance.getObjectPath(), "CIM_OrderedComponent", NULL, NULL,
842         NULL, $associatedBootOrder->[]);
843     &smSortInstancePaths($associatedBootOrder->[], "AssignedSequence",
844         "AscendingOrder", $orderedBootOrder->[]);
845     for #i in $orderedBootOrder[]
846     {

```

```

847     $smDisplayInstance($orderedBootOrder->[#i], propertylist2[]);
848     &smOpGetInstance($orderedBootOrder->[#i].PartComponent, NULL, $bootSource);
849     $smShowInstance($bootSource.GetObjectPath, propertylist3[]);
850 }
851 &smCommandCompleted($job);
852 &smEnd;
853 }
854 else if (4096 == #returnStatus) {
855     //job spawned, need to watch for it to finish
856     //while the jobstate is 4 ("Running")
857     while (4 == $instanceConcreteJob.JobState){<busy wait>}
858     //when job finishes, invoke GetError()
859     if (2 != $job.OperationalStatus) {
860         %InArguments[] = { }
861         %OutArguments[] = {newArgument("Job", $instanceConcreteJob.GetObjectPath())}
862         #Error = smOpInvokeMethod($job,
863             "GetError"
864             %InArguments,
865             %OutArguments,
866             #returncode);
867         //Method invocation failed, internal processing error
868         if ( (0 != #Error.code) || (0 != #returncode) ) {
869             //make an Error instance and associate with job for Operation
870             $OperationError = smNewInstance("CIM_Error");
871             //CIM_ERR_FAILED
872             $OperationError.CIMStatusCode = 1;
873             //Software Error
874             $OperationError.ErrorType = 4;
875             //Unknown
876             $OperationError.PerceivedSeverity = 0;
877             $OperationError.OwningEntity = DMTF:SMCLP;
878             $OperationError.MessageID = 0x00000009;
879             $OperationError.Message = "An internal software error has occurred.";
880             &smAddError($job, $OperationError);
881             &smMakeCommandStatus($job);
882             &smEnd;
883         } else {
884             //make command status
885             $joberror = %OutArguments["Error"];
886             &smMakeCommandExecutionFailed($job, {$joberror});
887         } //end if have CIM_Error from GetError()
888     } //embedded job not OK
889 } else {
890     //unspecified return code, generic failure.
891     $OperationError = smNewInstance("CIM_Error");
892     //CIM_ERR_FAILED
893     $OperationError.CIMStatusCode = 1;
894     //Other
895     $OperationError.ErrorType = 1;

```

```

896 //Low
897 $OperationError.PerceivedSeverity = 2;
898 $OperationError.OwningEntity = DMTF:SMCLP;
899 $OperationError.MessageID = 0x00000002;
900 $OperationError.Message = "Failed. No further information is available.";
901 &smAddError($job, $OperationError);
902 &smMakeCommandStatus($job);
903 &smEnd;
904 }

```

905 6.3.5 Show

906 The `show` verb is used to display information about instances of `CIM_BootConfigSetting`.
 907 Implementations shall support the use of the `show` verb with `CIM_BootConfigSetting`.

908 6.3.5.1 Show a Single Instance

909 This command form is used to display the information about a single instance of `CIM_BootConfigSetting`.

910 6.3.5.1.1 Command Form

```
911 show <CIM_BootConfigSetting single instance>
```

912 6.3.5.1.2 CIM Requirements

913 See `CIM_BootConfigSetting` in the "CIM Elements" section of the [Boot Control Profile](#) for the list of
 914 mandatory properties.

915 6.3.5.1.3 Behavior Requirements

916 6.3.5.1.3.1 Preconditions

917 `$instance` represents the instance of `CIM_BootSettingData`.

918 `#all` is true, if the "-all" option was specified with the command; otherwise, `#all` is false.

919 `#propertylist1[]` is an array of mandatory non-key property names for `CIM_BootConfigSetting`.

920 `#propertylist2[]` is an array of mandatory non-key property names for `CIM_OrderedComponent`.

921 `#propertylist3[]` is an array of mandatory non-key property names for `CIM_BootSourceSetting`.

922 6.3.5.1.3.2 Pseudo Code

```

923 if (false != #all) { #propertylist1[] = NULL; }
924 &smShowInstance ( $instance.getObjectPath(), #propertylist1[] );
925 // Show boot sources in order
926 &smOpReferences ( $instance.getObjectPath(), "CIM_OrderedComponent", NULL, NULL, NULL,
927 $associatedBootOrder->[] );
928 &smSortInstancePaths ( $associatedBootOrder->[], "AssignedSequence", "AscendingOrder",
929 $orderedBootOrder->[] );
930 for #i in $orderedBootOrder[]
931 {
932 $smDisplayInstance ( $orderedBootOrder->#[#i], #propertylist2[] );
933 &smOpGetInstance ( $orderedBootOrder->#[#i].PartComponent, NULL, $bootSource );
934 $smShowInstance ( $bootSource, #propertylist3[] );
935 }
936 &smEnd;

```

937 **6.3.5.2 Show Multiple Instances**

938 This command form applies the show verb to multiple instances of CIM_BootConfigSetting. This
 939 command form uses the UFsT as the target.

940 **6.3.5.2.1 Command Form**

```
941 show <CIM_BootConfigSetting multiple instances>
```

942 **6.3.5.2.2 CIM Requirements**

943 See CIM_BootConfigSetting in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
 944 mandatory properties.

945 **6.3.5.2.3 Behavior Requirements**

946 **6.3.5.2.3.1 Preconditions**

947 \$containerInstance represents the instance of CIM_ComputerSystem for which the
 948 CIM_BootConfigSetting instances are being displayed. The CIM_BootConfigSetting instances are
 949 associated to CIM_ComputerSystem via an instance of the CIM_ElementSettingData association.

950 #all is true, if the “-all” option was specified with the command; otherwise, #all is false.

951 #propertylist[] is an array of mandatory non-key property names.

952 **6.3.5.2.3.2 Pseudo Code**

```
953 if (false != #all) { #propertylist[] = NULL; }  

    954 &smShowInstances ( "CIM_BootConfigSetting", "CIM_ElementSettingData",  

    955     $containerInstance.getObjectPath(), #propertylist[] );  

    956 &smEnd;
```

957 **6.4 CIM_BootSettingData**

958 The cd and help verbs shall be supported as described in [DSP0216](#).

959 Table 4 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
 960 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
 961 target. Table 4 is for informational purposes only; in case of a conflict between Table 4 and requirements
 962 detailed in the following sections, the text detailed in the following sections supersedes the information in
 963 Table 4.

964 **Table 4 – Command Verb Requirements for CIM_BootSettingData**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	

Command Verb	Requirement	Comments
show	Shall	See 6.4.2.
start	Not supported	
stop	Not supported	

965 No mappings are defined for the following verbs for the specified target: create, delete, dump, exit,
 966 load, reset, set, start, and stop.

967 **6.4.1 Ordering of Results**

968 When results are returned for multiple instances of CIM_BootSettingData, implementations shall utilize
 969 the following algorithm to produce the natural (that is, default) ordering:

- 970 • Results for CIM_BootSettingData are unordered; therefore, no algorithm is defined.

971 **6.4.2 Show**

972 The `show` verb is used to display information about instances of CIM_BootSettingData. Implementations
 973 shall support the use of the `show` verb with CIM_BootSettingData.

974 **6.4.2.1 Show a Single Instance**

975 This command form is used to display the information about a single instance of CIM_BootSettingData.

976 **6.4.2.1.1 Command Form**

```
977 show <CIM_BootSettingData single instance>
```

978 **6.4.2.1.2 CIM Requirements**

979 See CIM_BootSettingData in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
 980 mandatory properties.

981 **6.4.2.1.3 Behavior Requirements**

982 **6.4.2.1.3.1 Preconditions**

983 In this section, `$instance` represents the instance of CIM_BootSettingData.

984 `#all` is true, if the “-all” option was specified with the command; otherwise, `#all` is false.

985 `#propertylist[]` is an array of mandatory non-key property names.

986 **6.4.2.1.3.2 Pseudo Code**

```
987 if (false != #all) { #propertylist[] = NULL; }
988 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
989 &smEnd;
```

990 **6.4.2.2 Show Multiple Instances**

991 This command form is used to display the information about multiple instances of
 992 CIM_BootConfigSetting. This command form corresponds to UFT-based selection within a scoping
 993 system.

994 **6.4.2.2.1 Command Form**

```
995 show <CIM_BootSettingData multiple instances>
```

996 **6.4.2.2.2 CIM Requirements**

997 See CIM_BootSettingData in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
998 mandatory properties.

999 **6.4.2.2.3 Behavior Requirements**

1000 **6.4.2.2.3.1 Preconditions**

1001 In this section, \$containerInstance represents the instance of CIM_BootConfigSetting or
1002 CIM_BootSourceSetting that contains the instance of CIM_BootSettingData.

1003 #all is true, if the “-all” option was specified with the command; otherwise, #all is false.

1004 #propertylist[] is an array of mandatory non-key property names.

1005 **6.4.2.2.3.2 Pseudo Code**

```
1006 if (false != #all) { #propertylist[] = NULL; }
1007 &smShowInstances ( "CIM_BootSettingData", "CIM_ConcreteComponent",
1008     $containerInstance.getObjectPath(), #propertylist[] );
1009 &smEnd;
```

1010 **6.5 CIM_BootSourceSetting**

1011 The cd and help verbs shall be supported as described in [DSP0216](#).

1012 Table 5 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
1013 class, and when appropriate, a cross-reference to the section detailing the mapping for the verb and
1014 target. Table 5 is for informational purposes only; in case of a conflict between Table 5 and requirements
1015 detailed in the following sections, the text detailed in the following sections supersedes the information in
1016 Table 5.

1017 **Table 5 – Command Verb Requirements for CIM_BootSourceSetting**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.5.2.
start	Not supported	
stop	Not supported	

1018 No mappings are defined for the following verbs for the specified target: create, delete, dump, exit,
1019 load, reset, start, and stop.

1020 6.5.1 Ordering of Results

1021 When results are returned for multiple instances of CIM_BootSourceSetting, implementations shall utilize
1022 the following algorithm to produce the natural (that is, default) ordering.

- 1023 • Results for CIM_BootSourceSetting are ordered based on the AssignedSequence property of
1024 the CIM_OrderedComponent used to associate the instance of CIM_BootSourceSetting with the
1025 instance of CIM_BootConfigSetting.
- 1026 • The order of display will be in increasing values of CIM_OrderedComponent.AssignedSequence
1027 property.
- 1028 • When the "-all" option is present on the command line, the instances of CIM_BootSourceSetting
1029 whose associated CIM_OrderedComponent.AssignedSequence property matches 0 (zero) shall
1030 be displayed. These instances of CIM_BootSourceSetting will be displayed following the
1031 CIM_BootSourceSetting instances with non-zero AssignedSequence values. These instances
1032 of CIM_BootSourceSetting are unordered and no algorithm is defined.

1033 6.5.2 Show

1034 The `show` verb is used to display information about instances of CIM_BootSourceSetting.
1035 Implementations shall support the use of the `show` verb with CIM_BootSourceSetting.

1036 6.5.2.1 Show a Single Instance

1037 This command form is used to display the information about a single instance of CIM_BootSourceSetting.

1038 6.5.2.1.1 Command Form

```
1039 show <CIM_BootSourceSetting single instance>
```

1040 6.5.2.1.2 CIM Requirements

1041 See CIM_BootSourceSetting in the "CIM Elements" section of the [Boot Control Profile](#) for the list of
1042 mandatory properties.

1043 6.5.2.1.3 Behavior Requirements

1044 6.5.2.1.3.1 Preconditions

1045 `$instance` represents the instance of CIM_BootSourceSetting.

1046 `#all` is true, if the "-all" option was specified with the command; otherwise, `#all` is false.

1047 `#propertylist[]` is an array of mandatory non-key property names.

1048 6.5.2.1.3.2 Pseudo Code

```
1049 if (false != #all) { #propertylist[] = NULL; }
1050 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
1051 // If the boot source is a logical identity to CIM_BootConfigSetting,
1052 // show the association.
1053 &smOpReferences ( $instance.getObjectPath(), "CIM_LogicalIdentity", "SystemElement",
1054 NULL, NULL, $reference[] );
1055 &smShowInstance ( $reference->[0]), NULL );
1056 &smEnd;
```

1057 6.5.2.2 Show Multiple Instances

1058 This command form is used to display the information about multiple instances of
1059 CIM_BootSourceSetting. This command form uses the UFsT as the target.

1060 6.5.2.2.1 Command Form

```
1061 show <CIM_BootSourceSetting multiple instances>
```

1062 6.5.2.2.2 CIM Requirements

1063 See CIM_BootSourceSetting in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
1064 mandatory properties.

1065 6.5.2.2.3 Behavior Requirements

1066 6.5.2.2.3.1 Preconditions

1067 \$containerInstance represents the instance of CIM_BootConfigSetting for which the
1068 CIM_BootSourceSetting instances are being displayed. The CIM_BootSourceSetting instances are
1069 associated to CIM_BootConfigSetting via an instance of the CIM_OrderedComponent association.

1070 #all is true, if the “-all” option was specified with the command; otherwise, #all is false.

1071 #propertylist[] is an array of mandatory non-key property names.

1072 6.5.2.2.3.2 Pseudo Code

```
1073 if (false != #all) { #propertylist[] = NULL; }
1074 &smOpReferences ( $containerInstance, "CIM_OrderedComponent", NULL, NULL, NULL,
1075   $associatedBootOrder->[] );
1076 &smSortInstancePaths ( $associatedBootOrder->[], "AssignedSequence", "AscendingOrder",
1077   $orderedBootOrder->[] );
1078 for #i in $orderedBootOrder[]
1079 {
1080     if ( $orderedBootOrder[#i].AssignedSequence != 0 )
1081     {
1082         $smDisplayInstance ( $orderedBootOrder->#[#i], propertylist2[] );
1083         &smOpGetInstance ( $orderedBootOrder->#[#i].PartComponent, NULL, $bootSource );
1084         $smShowInstance ( $bootSource.GetObjectPath, propertylist3[] );
1085     }
1086 }
1087 &smEnd;
```

1088 6.6 CIM_ConcreteComponent

1089 The cd and help verbs shall be supported as described in [DSP0216](#).

1090 Table 6 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
1091 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
1092 target. Table 6 is for informational purposes only; in case of a conflict between Table 6 and requirements
1093 detailed in the following sections, the text detailed in the following sections supersedes the information in
1094 Table 6.

1095

Table 6 – Command Verb Requirements for CIM_ConcreteComponent

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.6.2.
Start	Not supported	
Stop	Not supported	

1096 No mappings are defined for the following verbs for the specified target: create, delete, dump, exit,
 1097 load, reset, set, start, and stop.

1098 **6.6.1 Ordering of Results**

1099 When results are returned for multiple instances of CIM_ConcreteComponent, implementations shall
 1100 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1101 • Results for CIM_ConcreteComponent are unordered; therefore, no algorithm is defined.

1102 **6.6.2 Show**

1103 This section describes how to implement the show verb when applied to an instance of
 1104 CIM_ConcreteComponent. Implementations shall support the use of the show verb with
 1105 CIM_ConcreteComponent.

1106 **6.6.2.1 Show Multiple Instances – CIM_BootConfigSetting Reference**

1107 This command form is used when the show verb applies to multiple instances. This command form
 1108 corresponds to a show command issued against instances of CIM_ConcreteComponent where only one
 1109 reference is specified and the reference is to the instance of CIM_BootConfigSetting.

1110 **6.6.2.1.1 Command Form**

1111 `show <CIM_ConcreteComponent multiple instances>`

1112 **6.6.2.1.2 CIM Requirements**

1113 See CIM_ConcreteComponent in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
 1114 mandatory properties.

1115 **6.6.2.1.3 Behavior Requirements**

1116 **6.6.2.1.3.1 Preconditions**

1117 \$instance represents the instance of CIM_BootConfigSetting, which is referenced by
 1118 CIM_ConcreteComponent.

1119 **6.6.2.1.3.2 Pseudo Code**

```
1120 &smShowAssociationInstances ( "CIM_ConcreteComponent", $instance.getObjectPath() );
1121 &smEnd;
```

1122 **6.6.2.2 Show Multiple Instances – CIM_BootSourceSetting Reference**

1123 This command form is used when the `show` verb applies to multiple instances. This command form
 1124 corresponds to a `show` command issued against instances of `CIM_ConcreteComponent` where only one
 1125 reference is specified and the reference is to the instance of `CIM_BootSourceSetting`.

1126 **6.6.2.2.1 Command Form**

```
1127 show <CIM_ConcreteComponent multiple instances>
```

1128 **6.6.2.2.2 CIM Requirements**

1129 See `CIM_ConcreteComponent` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
 1130 mandatory properties.

1131 **6.6.2.2.3 Behavior Requirements**1132 **6.6.2.2.3.1 Preconditions**

1133 `$instance` represents the instance of `CIM_BootSourceSetting`, which is referenced by
 1134 `CIM_ConcreteComponent`.

1135 **6.6.2.2.3.2 Pseudo Code**

```
1136 &smShowAssociationInstances ( "CIM_ConcreteComponent", $instance.getObjectPath() );
1137 &smEnd;
```

1138 **6.6.2.3 Show a Single Instance – CIM_BootSettingData Reference**

1139 This command form is used when the `show` verb applies to instances of `CIM_ConcreteComponent` where
 1140 only one reference is specified and the reference is to an instance of a concrete subclass of
 1141 `CIM_BootSettingData`.

1142 The [Boot Control Profile](#) imposes a cardinality of 1 on `CIM_ConcreteComponent.GroupComponent`,
 1143 which references an instance of `CIM_BootConfigSetting`. Therefore, for a given instance of a concrete
 1144 subclass of `CIM_BootSettingData`, a single instance of `CIM_BootConfigSetting` is found.

1145 **6.6.2.3.1 Command Form**

```
1146 show <CIM_ConcreteComponent single instance>
```

1147 **6.6.2.3.2 CIM Requirements**

1148 See `CIM_ConcreteComponent` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
 1149 mandatory properties.

1150 **6.6.2.3.3 Behavior Requirements**1151 **6.6.2.3.3.1 Preconditions**

1152 `$instance` represents the instance a concrete subclass of `CIM_BootSettingData`.

1153 6.6.2.3.3.2 Pseudo Code

```

1154 &smOpReferences ( $instance.getObjectPath(), "CIM_ConcreteComponent", "PartComponent",
1155     NULL, NULL, $reference[] );
1156 &smShowInstance ( $reference->[0]), NULL );
1157 &smEnd;

```

1158 6.6.2.4 Show Multiple Instance – CIM_SettingData Reference

1159 This command form is used when the `show` verb applies to instances of `CIM_ConcreteComponent` where
 1160 only one reference is specified and the reference is to an instance of a concrete subclass of
 1161 `CIM_SettingData`.

1162 6.6.2.4.1 Command Form

```

1163 show <CIM_ConcreteComponent multiple instances>

```

1164 6.6.2.4.2 CIM Requirements

1165 See `CIM_ConcreteComponent` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
 1166 mandatory properties.

1167 6.6.2.4.3 Behavior Requirements

1168 6.6.2.4.3.1 Preconditions

1169 `$instance` represents the instance of a concrete subclass of `CIM_SettingData`.

1170 6.6.2.4.3.2 Pseudo Code

```

1171 &smShowAssociationInstances ( "CIM_ConcreteComponent", $instance.getObjectPath() );
1172 &smEnd;

```

1173 6.6.2.5 Show a Single Instance – Both References

1174 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 1175 a `show` command issued against `CIM_ConcreteComponent` where both references are specified and
 1176 therefore the desired instance is unambiguously identified.

1177 6.6.2.5.1 Command Form

```

1178 show <CIM_ConcreteComponent single instance>

```

1179 6.6.2.5.2 CIM Requirements

1180 See `CIM_ConcreteComponent` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
 1181 mandatory properties.

1182 6.6.2.5.3 Behavior Requirements

1183 6.6.2.5.3.1 Preconditions

1184 `$instanceA` represents the instance of `CIM_BootConfigSetting` or the instance of
 1185 `CIM_BootSourceSetting` which is referenced by `CIM_ConcreteComponent`.

1186 `$instanceB` represents the instance of `CIM_BootSettingData` or the instance of `CIM_SettingData` which
 1187 is referenced by `CIM_ConcreteComponent`.

1188 **6.6.2.5.3.2 Pseudo Code**

```
1189 &smShowAssociationInstance ( "CIM_ConcreteComponent", $instanceA.getObjectPath(),
1190     $instanceB.getObjectPath() );
1191 &smEnd;
```

1192 **6.7 CIM_ConcreteDependency**

1193 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

1194 Table 7 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
 1195 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
 1196 target. Table 7 is for informational purposes only; in case of a conflict between Table 7 and requirements
 1197 detailed in the following sections, the text detailed in the following sections supersedes the information in
 1198 Table 7.

1199 **Table 7 – Command Verb Requirements for CIM_ConcreteDependency**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.7.2.
Start	Not supported	
Stop	Not supported	

1200 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `exit`,
 1201 `load`, `reset`, `set`, `start`, and `stop`.

1202 **6.7.1 Ordering of Results**

1203 When results are returned for multiple instances of `CIM_ConcreteDependency`, implementations shall
 1204 utilize the following algorithm to produce the natural (that is, default) ordering.

- 1205 • Results for `CIM_ConcreteDependency` are unordered; therefore, no algorithm is defined.

1206 **6.7.2 Show**

1207 This section describes how to implement the `show` verb when applied to an instance of
 1208 `CIM_ConcreteDependency`. Implementations shall support the use of the `show` verb with
 1209 `CIM_ConcreteDependency`.

1210 **6.7.2.1 Show Multiple Instances – CIM_BootSourceSetting Reference**

1211 This command form is used when the `show` verb applies to multiple instances. This command form
 1212 corresponds to a `show` command issued against instances of `CIM_ConcreteDependency` where only one
 1213 reference is specified and the reference is to the instance of `CIM_BootSourceSetting`.

1214 6.7.2.1.1 Command Form

```
1215 show <CIM_ConcreteDependency multiple instances>
```

1216 6.7.2.1.2 CIM Requirements

1217 See CIM_ConcreteDependency in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
1218 mandatory properties.

1219 6.7.2.1.3 Behavior Requirements

1220 6.7.2.1.3.1 Preconditions

1221 \$instance represents the instance of CIM_BootSourceSetting which is referenced by
1222 CIM_ConcreteDependency.

1223 6.7.2.1.3.2 Pseudo Code

```
1224 &smShowAssociationInstances ( "CIM_ConcreteDependency", $instance.getObjectPath() );  
1225 &smEnd;
```

1226 6.7.2.2 Show Multiple Instances – CIM_LogicalDevice Reference

1227 This command form is used when the `show` verb applies to instances of CIM_ConcreteDependency
1228 where only one reference is specified and the reference is to the instance of a concrete subclass of
1229 CIM_LogicalDevice.

1230 6.7.2.2.1 Command Form

```
1231 show <CIM_ConcreteDependency multiple instances>
```

1232 6.7.2.2.2 CIM Requirements

1233 See CIM_ConcreteDependency in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
1234 mandatory properties.

1235 6.7.2.2.3 Behavior Requirements

1236 6.7.2.2.3.1 Preconditions

1237 \$instance represents the instance of CIM_LogicalDevice which is referenced by
1238 CIM_ConcreteDependency.

1239 6.7.2.2.3.2 Pseudo Code

```
1240 &smShowAssociationInstances ( "CIM_ConcreteDependency", $instance.getObjectPath() );  
1241 &smEnd;
```

1242 6.7.2.3 Show a Single Instance – Both References

1243 This command form is for the `show` verb applied to a single instance. This command form corresponds to
1244 a `show` command issued against CIM_ConcreteDependency where both references are specified and
1245 therefore the desired instance is unambiguously identified.

1246 6.7.2.3.1 Command Form

```
1247 show <CIM_ConcreteDependency single instance>
```

1248 **6.7.2.3.2 CIM Requirements**

1249 See CIM_ConcreteDependency in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
1250 mandatory properties.

1251 **6.7.2.3.3 Behavior Requirements**

1252 **6.7.2.3.3.1 Preconditions**

1253 \$instanceA represents the instance of CIM_BootSourceSetting which is referenced by
1254 CIM_ConcreteDependency.

1255 \$instanceB represents the instance of CIM_LogicalDevice which is referenced by
1256 CIM_ConcreteDependency.

1257 **6.7.2.3.3.2 Pseudo Code**

```
1258 &smShowAssociationInstance ( "CIM_ConcreteDependency", $instanceA.getObjectPath(),
1259     $instanceB.getObjectPath() );
1260 &smEnd;
```

1261 **6.8 CIM_ElementCapabilities**

1262 The cd, help, version, and exit verbs shall be supported as described in [DSP0216](#).

1263 Table 8 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
1264 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
1265 target. Table 8 is for informational purposes only; in case of a conflict between Table 8 and requirements
1266 detailed in the following sections, the text detailed in the following sections supersedes the information in
1267 Table 8.

1268 **Table 8 – Command Verb Requirements for CIM_ElementCapabilities**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.8.2.
Start	Not supported	
Stop	Not supported	

1269 No mapping is defined for the following verbs for the specified target: create, delete, dump, load,
1270 reset, set, start, and stop.

1271 **6.8.1 Ordering of Results**

1272 When results are returned for multiple instances of CIM_ElementCapabilities, implementations shall
1273 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1274 • Results for CIM_ElementCapabilities are unordered; therefore, no algorithm is defined.

1275 **6.8.2 Show**

1276 This section describes how to implement the `show` verb when applied to an instance of
 1277 `CIM_ElementCapabilities`. Implementations shall support the use of the `show` verb with
 1278 `CIM_ElementCapabilities`.

1279 **6.8.2.1.1 Show a Single Instance – CIM_BootService Reference**

1280 This command form is used to apply the `show` verb to an instance of `CIM_ElementCapabilities` where
 1281 only one reference is specified and the reference is to the instance of `CIM_BootService`.

1282 **6.8.2.1.2 Command Form**

```
1283 show <CIM_ElementCapabilities single instance>
```

1284 **6.8.2.1.3 CIM Requirements**

1285 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
 1286 mandatory properties.

1287 **6.8.2.1.4 Behavior Requirements**

1288 **6.8.2.1.4.1 Preconditions**

1289 `$instance` represents the instance of a `CIM_BootService`, which is referenced by
 1290 `CIM_ElementCapabilities`.

1291 `#all` is true, if the “-all” option was specified with the command; otherwise, `#all` is false.

1292 `#propertylist[]` is an array of mandatory non-key property names.

1293 **6.8.2.1.4.2 Pseudo Code**

```
1294 if ( false != #all) { #propertylist[] = NULL; }
1295 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath(),
1296     #propertylist[] );
1297 &smEnd;
```

1298 **6.8.2.2 Show Command Form for Multiple Instances – CIM_BootServiceCapabilities Reference**

1299 This command form is used to show a single instance of `CIM_ElementCapabilities`. This command form
 1300 corresponds to a `show` command issued against a single instance of `CIM_ElementCapabilities` where
 1301 only one reference is specified and the reference is to the instance of `CIM_BootServiceCapabilities`.

1302 **6.8.2.2.1 Command Form**

```
1303 show <CIM_ElementCapabilities multiple instances>
```

1304 **6.8.2.2.2 CIM Requirements**

1305 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
 1306 mandatory properties.

1307 6.8.2.2.3 Behavior Requirements**1308 6.8.2.2.3.1 Preconditions**

1309 \$instance represents the instance of a CIM_BootCapabilities, which is referenced by
1310 CIM_ElementCapabilities.

1311 #all is true, if the “-all” option was specified with the command; otherwise, #all is false.

1312 #propertylist[] is an array of mandatory non-key property names.

1313 6.8.2.2.3.2 Pseudo Code

```
1314 if ( false != #all) { #propertylist[] = NULL; }  
1315 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath(),  
1316     #propertylist[]);  
1317 &smEnd;
```

**1318 6.8.2.3 Show Command Form for a Single Instance Target – CIM_BootService and
1319 CIM_BootServiceCapabilities References**

1320 This command form is for the show verb applied to a single instance. This command form corresponds to
1321 a show command issued against CIM_ElementCapabilities where both references are specified and
1322 therefore the desired instance is unambiguously identified.

1323 6.8.2.3.1 Command Form

```
1324 show <CIM_ElementCapabilities single instance>
```

1325 6.8.2.3.2 CIM Requirements

1326 See CIM_ElementCapabilities in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
1327 mandatory properties.

1328 6.8.2.3.3 Behavior Requirements**1329 6.8.2.3.3.1 Preconditions**

1330 \$instanceA represents the instance of a CIM_BootService, which is referenced by
1331 CIM_ElementCapabilities.

1332 \$instanceB represents the instance of a CIM_BootServiceCapabilities, which is referenced by
1333 CIM_ElementCapabilities.

1334 #all is true, if the “-all” option was specified with the command; otherwise, #all is false.

1335 #propertylist[] is an array of mandatory non-key property names.

1336 6.8.2.3.3.2 Pseudo Code

```
1337 if ( false != #all) { #propertylist[] = NULL; }  
1338 &smShowAssociationInstance ( "CIM_ElementCapabilities", $instanceA.getObjectPath(),  
1339     $instanceB.getObjectPath(), #propertylist[] );  
1340 &smEnd;
```


1341 **6.9 CIM_ElementSettingData**

1342 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

1343 Table 9 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
 1344 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
 1345 target. Table 9 is for informational purposes only; in case of a conflict between Table 9 and requirements
 1346 detailed in the following sections, the text detailed in the following sections supersedes the information in
 1347 Table 9.

1348 **Table 9 – Command Verb Requirements for CIM_ElementSettingData**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	May	See 6.9.2.
show	Shall	See 6.9.3.
start	Not supported	
stop	Not supported	

1349 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `exit`,
 1350 `load`, `reset`, `start`, and `stop`.

1351 **6.9.1 Ordering of Results**

1352 When results are returned for multiple instances of `CIM_ElementSettingData`, implementations shall
 1353 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1354 • Results for `CIM_ElementSettingData` are unordered; therefore, no algorithm is defined.

1355 **6.9.2 Set**

1356 This section describes how to implement the `set` verb when applied to an instance of
 1357 `CIM_ElementSettingData`. Implementations may support the use of the `set` verb for an instance of
 1358 `CIM_ElementSettingData` which references an instance of `CIM_BootConfigSetting`.

1359 **6.9.2.1 Set of IsNext Property**

1360 This command form is for when the `set` verb is used to change the value of the `IsNext` property of an
 1361 instance of `CIM_ElementSettingData` that associates an instance of `CIM_ComputerSystem` with an
 1362 instance of `CIM_BootConfigSetting`. The valid input values are 1 (Is Next), 2 (Is Not Next), or 3 (Is Next
 1363 for Single Use).

1364 **6.9.2.1.1 Command Form**

1365 `set <CIM_ElementSettingData single instance> IsNext=<propertyvalue>`

1366 **6.9.2.1.2 CIM Requirements**

1367 See CIM_ElementSettingData in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
1368 mandatory properties.

1369 **6.9.2.1.3 Behavior Requirements**1370 **6.9.2.1.3.1 Preconditions**

1371 \$instance represents the instance of CIM_ElementSettingData.

1372 The property value specified in the command is one of the following valid values: 1 (Is Next), 2 (Is Not
1373 Next), or 3 (Is Next for Single Use).

1374 #intpropertyvalue is the integer value of the <propertyvalue>.

1375 **6.9.2.1.3.2 Pseudo Code**

```

1376 if (#intpropertyvalue != 1 && #intpropertyvalue != 2 && #intpropertyvalue != 3) {
1377     $OperationError = smNewInstance("CIM_Error");
1378     //CIM_ERR_FAILED
1379     $OperationError.CIMStatusCode = 1;
1380     //Software Error
1381     $OperationError.ErrorType = 4;
1382     //Unknown
1383     $OperationError.PerceivedSeverity = 0;
1384     $OperationError.OwningEntity = DMTF:SMCLP;
1385     $OperationError.MessageID = 0x0000000E;
1386     $OperationError.Message = "The value specified for the {1} property is not valid.";
1387     $OperationError.MessageArguments = { "IsNext" };
1388     &smAddError($job, $OperationError);
1389     &smMakeCommandStatus($job);
1390     &smEnd;
1391 }
1392 #propertyName[] = "IsNext"
1393 #propertyValues[] = #intpropertyvalue
1394 &smSetInstance($instance, #propertyName, #propertyValues);
1395 &smEnd;

```

1396 **6.9.2.2 Set of IsCurrent Property**

1397 This command form is for when the `set` verb is used to change the value of the `IsCurrent` property of an
1398 instance of `CIM_ElementSettingData` that associates an instance of `CIM_ComputerSystem` with an
1399 instance of `CIM_BootConfigSetting`. The valid input value is 1 (Is Current).

1400 This command form causes the Boot Configurable System to start the boot process, which applies the
1401 Next Boot Configuration or Next Single Use Boot Configuration. The command may be used when the
1402 boot process is started automatically as part of the system start or reset.

1403 **6.9.2.2.1 Command Form**

```
1404 set <CIM_ElementSettingData single instance> IsCurrent=<propertyvalue>
```

1405 **6.9.2.2.2 CIM Requirements**

1406 See CIM_ElementSettingData in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
1407 mandatory properties.

1408 **6.9.2.2.3 Behavior Requirements**1409 **6.9.2.2.3.1 Preconditions**

1410 \$instance represents the instance of CIM_ElementSettingData.

1411 The property value specified in the command is the valid value, 1 (Is Current).

1412 #intpropertyvalue is the integer value of the <propertyvalue>.

1413 **6.9.2.2.3.2 Pseudo Code**

```

1414 if (#intpropertyvalue != 1) {
1415     $OperationError = smNewInstance("CIM_Error");
1416     //CIM_ERR_FAILED
1417     $OperationError.CIMStatusCode = 1;
1418     //Software Error
1419     $OperationError.ErrorType = 4;
1420     //Unknown
1421     $OperationError.PerceivedSeverity = 0;
1422     $OperationError.OwningEntity = DMTF:SMCLP;
1423     $OperationError.MessageID = 0x0000000E;
1424     $OperationError.Message = "The value specified for the {1} property is not valid.";
1425     $OperationError.MessageArguments = { "IsCurrent" };
1426     &smAddError($job, $OperationError);
1427     &smMakeCommandStatus($job);
1428     &smEnd;
1429 }
1430 // Try to find the CIM_BootService, take the first instance found
1431 $Services[] = smOpAssociators($instance.ManagedElement,
1432     "CIM_ServiceAffectsElement", "CIM_BootService", NULL, NULL);
1433 $Service-> = $Services[0].getObjectPath();
1434 //Step 6, build parameter lists for method invocation
1435 %InArguments[] = {
1436     newArgument("BootConfigurableSystem", $instance.ManagedElement)
1437     newArgument("ApplyBootConfig", $instance.SettingData)
1438 };
1439 %OutArguments[] = { newArgument("Job", instanceConcreteJob.getObjectPath() ) };
1440 //step 7, invoke method
1441 #returnStatus = smOpInvokeMethod ($Service->,
1442     "ApplyBootConfigSetting",
1443     %InArguments[],
1444     %OutArguments[]);
1445 //step 8, process return code to CLP Command Status
1446 if (0 != #Error.code) {
1447     //method invocation failed
1448     if ( (null != #Error.$error) && (null != #Error.$error[0]) ) {
1449         //if the method invocation contains an embedded error
1450         //use it for the Error for the overall job

```

```

1451     &smAddError($job, #Error.$error[0]);
1452     &smMakeCommandStatus($job);
1453     &smEnd;
1454 }
1455 else if ( 17 == #Error.code ) {
1456     //17 - CIM_ERR_METHOD_NOT_FOUND
1457     // The specified extrinsic method does not exist.
1458     $OperationError = smNewInstance("CIM_Error");
1459     // CIM_ERR_METHOD_NOT_FOUND
1460     $OperationError.CIMStatusCode = 17;
1461     //Software Error
1462     $OperationError.ErrorType = 10;
1463     //Unknown
1464     $OperationError.PerceivedSeverity = 0;
1465     $OperationError.OwningEntity = DMTF:SMCLP;
1466     $OperationError.MessageID = 0x00000001;
1467     $OperationError.Message = "Operation is not supported."
1468     &smAddError($job, $OperationError);
1469     &smMakeCommandStatus($job);
1470     &smEnd;
1471 }
1472 else {
1473     //operation failed, but no detailed error instance, need to make one up
1474     //make an Error instance and associate with job for Operation
1475     $OperationError = smNewInstance("CIM_Error");
1476     //CIM_ERR_FAILED
1477     $OperationError.CIMStatusCode = 1;
1478     //Software Error
1479     $OperationError.ErrorType = 4;
1480     //Unknown
1481     $OperationError.PerceivedSeverity = 0;
1482     $OperationError.OwningEntity = DMTF:SMCLP;
1483     $OperationError.MessageID = 0x00000009;
1484     $OperationError.Message = "An internal software error has occurred.";
1485     &smAddError($job, $OperationError);
1486     &smMakeCommandStatus($job);
1487     &smEnd;
1488 }
1489 }//if CIM op failed
1490 else if (0 == #returnStatus) {
1491     //completed successfully
1492     &smCommandCompleted($job);
1493     &smEnd;
1494 }
1495 else if (4096 == #returnStatus) {
1496     //job spawned, need to watch for it to finish
1497     //while the jobstate is "Running"
1498     while (4 == $instanceConcreteJob.JobState){<busy wait>}
1499     if (2 != $job.OperationalStatus) {
1500         %InArguments[] = { }
1501         %OutArguments[] = {newArgument("Job", $instanceConcreteJob.getObjectPath())}

```

```

1502     #Error = smOpInvokeMethod($job,
1503         "GetError"
1504         %InArguments,
1505         %OutArguments,
1506         #returncode);
1507     //Method invocation failed, internal processing error
1508     if ( (0 != #Error.code) || (0 != #returncode) ) {
1509     //make an Error instance and associate with job for Operation
1510         $OperationError = smNewInstance("CIM_Error");
1511         //CIM_ERR_FAILED
1512         $OperationError.CIMStatusCode = 1;
1513         //Software Error
1514         $OperationError.ErrorType = 4;
1515         //Unknown
1516         $OperationError.PerceivedSeverity = 0;
1517         $OperationError.OwningEntity = DMTF:SMCLP;
1518         $OperationError.MessageID = 0x00000009;
1519         $OperationError.Message = "An internal software error has occurred.";
1520         &smAddError($job, $OperationError);
1521         &smMakeCommandStatus($job);
1522         &smEnd;
1523     }
1524     else {
1525         //make command status
1526         $joberror = %OutArguments["Error"];
1527         &smMakeCommandExecutionFailed($job, {$joberror});
1528     } //end if have CIM_Error from GetError()
1529 } //embedded job not OK
1530 }
1531 else {
1532     //unspecified return code, generic failure
1533     $OperationError = smNewInstance("CIM_Error");
1534     //CIM_ERR_FAILED
1535     $OperationError.CIMStatusCode = 1;
1536     //Other
1537     $OperationError.ErrorType = 1;
1538     //Low
1539     $OperationError.PerceivedSeverity = 2;
1540     $OperationError.OwningEntity = DMTF:SMCLP;
1541     $OperationError.MessageID = 0x00000002;
1542     $OperationError.Message = "Failed. No further information is available.";
1543     &smAddError($job, $OperationError);
1544     &smMakeCommandStatus($job);
1545     &smEnd;
1546 }

```

1547 6.9.3 Show

1548 This section describes how to implement the `show` verb when applied to an instance of
1549 `CIM_ElementSettingData`. Implementations shall support the use of the `show` verb with
1550 `CIM_ElementSettingData`.

1551 6.9.3.1 Show a Single Instance – CIM_BootConfigSetting Reference

1552 This command form is used when the `show` verb applies to instances of `CIM_ElementSettingData` where
1553 only one reference is specified and the reference is to an instance of `CIM_BootConfigSetting`.

1554 The [Boot Control Profile](#) imposes a cardinality of 1 on `CIM_ElementSettingData.ManagedElement`, which
1555 references an instance of `CIM_ComputerSystem`. Therefore, for a given instance of
1556 `CIM_BootConfigSetting`, a single instance of `CIM_ComputerSystem` is found.

1557 6.9.3.1.1 Command Form

```
1558 show <CIM_ElementSettingData single instance>
```

1559 6.9.3.1.2 CIM Requirements

1560 See `CIM_ElementSettingData` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
1561 mandatory properties.

1562 6.9.3.1.3 Behavior Requirements**1563 6.9.3.1.3.1 Preconditions**

1564 `$instance` represents the instance of `CIM_BootConfigSetting`, which is referenced by
1565 `CIM_ElementSettingData`.

1566 `#all` is true, if the “-all” option was specified with the command; otherwise, `#all` is false.

1567 `#propertylist[]` is an array of mandatory non-key property names.

1568 6.9.3.1.3.2 Pseudo Code

```
1569 if (false != #all) { #propertylist[] = NULL; }  
1570 &smShowAssociationInstances ( "CIM_ElementSettingData", $instance.getObjectPath(),  
1571     #propertylist[] );  
1572 &smEnd;
```

1573 6.9.3.2 Show Multiple Instance – CIM_ComputerSystem Reference

1574 This command form is used when the `show` verb applies to instances of `CIM_ElementSettingData` where
1575 only one reference is specified and the reference is to the instance of `CIM_ComputerSystem`.

1576 6.9.3.2.1 Command Form

```
1577 show <CIM_ElementSettingData multiple instances>
```

1578 6.9.3.2.2 CIM Requirements

1579 See `CIM_ElementSettingData` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
1580 mandatory properties.

1581 6.9.3.2.3 Behavior Requirements**1582 6.9.3.2.3.1 Preconditions**

1583 `$instance` represents the instance of `CIM_ComputerSystem`, which is referenced by
1584 `CIM_ElementSettingData`.

1585 `#all` is true, if the “-all” option was specified with the command; otherwise, `#all` is false.

1586 `#propertylist[]` is an array of mandatory non-key property names.

1587 **6.9.3.2.3.2 Pseudo Code**

```

1588 if (false != #all) { #propertylist[] = NULL; }
1589 &smShowAssociationInstances ( "CIM_ElementSettingData", $instance.getObjectPath(),
1590     #propertylist[] );
1591 &smEnd;

```

1592 **6.9.3.3 Show a Single Instance – Both References**

1593 This command form is used when the `show` verb applies to a single instance. This command form
 1594 corresponds to a `show` command issued against `CIM_ElementSettingData` where both references are
 1595 specified and therefore the desired instance is unambiguously identified.

1596 **6.9.3.3.1 Command Form**

```

1597 show <CIM_ElementSettingData single instance>

```

1598 **6.9.3.3.2 CIM Requirements**

1599 See `CIM_ElementSettingData` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
 1600 mandatory properties.

1601 **6.9.3.3.3 Behavior Requirements**1602 **6.9.3.3.3.1 Preconditions**

1603 `$instanceA` represents the referenced instance of `CIM_ComputerSystem` through
 1604 `CIM_ElementSettingData` association.

1605 `$instanceB` represents the other instance of `CIM_BootConfigSetting` which is referenced by
 1606 `CIM_ElementSettingData`.

1607 `#all` is true, if the “-all” option was specified with the command; otherwise, `#all` is false.

1608 `#propertylist[]` is an array of mandatory non-key property names.

1609 **6.9.3.3.3.2 Pseudo Code**

```

1610 if (false != #all) { #propertylist[] = NULL; }
1611 &smShowAssociationInstance ( "CIM_ElementSettingData", $instanceA.getObjectPath(),
1612     $instanceB.getObjectPath(), #propertylist[] );
1613 &smEnd;

```

1614 **6.10 CIM_HostedService**

1615 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

1616 Table 10 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
 1617 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
 1618 target. Table 10 is for informational purposes only; in case of a conflict between Table 10 and
 1619 requirements detailed in the following sections, the text detailed in the following sections supersedes the
 1620 information in Table 10.

1621

Table 10 – Command Verb Requirements for CIM_HostedService

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.10.2.
Start	Not supported	
Stop	Not supported	

1622 No mappings are defined for the following verbs for the specified target: create, delete, dump, exit,
 1623 load, reset, set, start, and stop.

1624 **6.10.1 Ordering of Results**

1625 When results are returned for multiple instances of CIM_HostedService, implementations shall utilize the
 1626 following algorithm to produce the natural (that is, default) ordering:

- 1627 • Results for CIM_HostedService are unordered; therefore, no algorithm is defined.

1628 **6.10.2 Show**

1629 This section describes how to implement the show verb when applied to an instance of
 1630 CIM_HostedService. Implementations shall support the use of the show verb with CIM_HostedService.

1631 **6.10.2.1 Show Multiple Instances – CIM_ComputerSystem Reference**

1632 This command form applies the show verb to multiple instances. This command form corresponds to a
 1633 show command issued against instances of CIM_HostedService where only one reference is specified
 1634 and the reference is to an instance of CIM_ComputerSystem.

1635 **6.10.2.1.1 Command Form**

```
1636 show <CIM_HostedService multiple instances>
```

1637 **6.10.2.1.2 CIM Requirements**

1638 See CIM_HostedService in the “CIM Elements” section of the [Boot Control Profile](#) for the list of mandatory
 1639 properties.

1640 **6.10.2.1.3 Behavior Requirements**

1641 **6.10.2.1.3.1 Preconditions**

1642 \$instance represents the instance of CIM_ComputerSystem, which is referenced by
 1643 CIM_HostedService.

1644 **6.10.2.1.3.2 Pseudo Code**

```
1645 &smShowAssociationInstances ( "CIM_HostedService", $instance.getObjectPath() );
1646 &smEnd;
```

1647 **6.10.2.2 Show a Single Instance – CIM_BootService Reference**

1648 This command form is used when the `show` verb applies to instances of `CIM_HostedService` where only
 1649 one reference is specified and the reference is to an instance of `CIM_BootService`.

1650 The [Boot Control Profile](#) imposes a cardinality of 1 on `CIM_HostedService.Dependent`, which references
 1651 an instance of `CIM_ComputerSystem`. Therefore, for a given instance of `CIM_BootService`, a single
 1652 instance of `CIM_HostedService` is found.

1653 **6.10.2.2.1 Command Form**

```
1654 show <CIM_HostedService single instance>
```

1655 **6.10.2.2.2 CIM Requirements**

1656 See `CIM_HostedService` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of mandatory
 1657 properties.

1658 **6.10.2.2.3 Behavior Requirements**1659 **6.10.2.2.3.1 Preconditions**

1660 `$instance` represents the instance of `CIM_BootService`, which is referenced by `CIM_HostedService`.

1661 **6.10.2.2.3.2 Pseudo Code**

```
1662 &smOpReferences ( $instance.getObjectPath(), "CIM_HostedService", "Antecedent", NULL,
1663 NULL, $references[] );
1664 &smShowInstance ( $references->[0]), NULL );
1665 &smEnd;
```

1666 **6.10.2.3 Show a Single Instance – Both References**

1667 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 1668 a `show` command issued against `CIM_HostedService` where both references are specified and therefore
 1669 the desired instance is unambiguously identified.

1670 **6.10.2.3.1 Command Form**

```
1671 show <CIM_HostedService single instance>
```

1672 **6.10.2.3.2 CIM Requirements**

1673 See `CIM_HostedService` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of mandatory
 1674 properties.

1675 **6.10.2.3.3 Behavior Requirements**

1676 **6.10.2.3.3.1 Preconditions**

1677 \$instanceA represents the referenced instance of CIM_ComputerSystem through CIM_HostedService
1678 association.

1679 \$instanceB represents the other instance of CIM_BootService which is referenced by
1680 CIM_HostedService.

1681 **6.10.2.3.3.2 Pseudo Code**

```
1682 &smShowAssociationInstance ( "CIM_HostedService", $instanceA.getObjectPath(),
1683     $instanceB.getObjectPath() );
1684 &smEnd;
```

1685 **6.11 CIM_LogicalIdentity**

1686 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

1687 Table 11 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
1688 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
1689 target. Table 11 is for informational purposes only; in case of a conflict between Table 11 and
1690 requirements detailed in the following sections, the text detailed in the following sections supersedes the
1691 information in Table 11.

1692 **Table 11 – Command Verb Requirements for CIM_LogicalIdentity**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.11.2
Start	Not supported	
Stop	Not supported	

1693 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `exit`,
1694 `load`, `reset`, `set`, `start`, and `stop`.

1695 **6.11.1 Ordering of Results**

1696 When results are returned for multiple instances of CIM_LogicalIdentity, implementations shall utilize the
1697 following algorithm to produce the natural (that is, default) ordering:

- 1698 • Results for CIM_LogicalIdentity are unordered; therefore, no algorithm is defined.

1699 **6.11.2 Show**

1700 This section describes how to implement the `show` verb when applied to an instance of
1701 CIM_LogicalIdentity. Implementations shall support the use of the `show` verb with CIM_LogicalIdentity.

1702 6.11.2.1 Show a Single Instance – CIM_BootSourceSetting Reference

1703 This command form is used when the `show` verb applies to instances of `CIM_LogicalIdentity` where only
1704 one reference is specified and the reference is to an instance of `CIM_BootSourceSetting`.

1705 The [Boot Control Profile](#) imposes a cardinality of zero or 1 on `CIM_LogicalIdentity.SameElement`, which
1706 references an instance of `CIM_BootConfigSetting`. Therefore, for a given instance of
1707 `CIM_BootSourceSetting`, a single instance of `CIM_LogicalIdentity` is found.

1708 6.11.2.1.1 Command Form

```
1709 show <CIM_LogicalIdentity single instance>
```

1710 6.11.2.1.2 CIM Requirements

1711 See `CIM_LogicalIdentity` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of mandatory
1712 properties.

1713 6.11.2.1.3 Behavior Requirements

1714 6.11.2.1.3.1 Preconditions

1715 `$instance` represents the instance of `CIM_BootSourceSetting`, which is referenced by
1716 `CIM_LogicalIdentity`.

1717 `#all` is true, if the “-all” option was specified with the command; otherwise, `#all` is false.

1718 `#propertylist[]` is an array of mandatory non-key property names.

1719 6.11.2.1.3.2 Pseudo Code

```
1720 if (false != #all) { #propertylist[] = NULL; }
1721 &smShowAssociationInstances ( "CIM_LogicalIdentity", $instance.getObjectPath(),
1722     #propertylist[] );
1723 &smEnd;
```

1724 6.11.2.2 Show a Single Instance – CIM_BootConfigSetting Reference

1725 This command form is used when the `show` verb applies to instances of `CIM_LogicalIdentity` where only
1726 one reference is specified and the reference is to an instance of `CIM_BootConfigSetting`.

1727 The [Boot Control Profile](#) imposes a cardinality of zero or 1 on `CIM_LogicalIdentity.SystemElement`, which
1728 references an instance of `CIM_BootSourceSetting`. Therefore, for a given instance of
1729 `CIM_BootConfigSetting`, a single instance of `CIM_LogicalIdentity` is found.

1730 6.11.2.2.1 Command Form

```
1731 show <CIM_LogicalIdentity single instance>
```

1732 6.11.2.2.2 CIM Requirements

1733 See `CIM_LogicalIdentity` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of mandatory
1734 properties.

1735 **6.11.2.2.3 Behavior Requirements**1736 **6.11.2.2.3.1 Preconditions**

1737 \$instance represents the instance of CIM_BootConfigSetting, which is referenced by
1738 CIM_LogicalIdentity.

1739 #all is true, if the “-all” option was specified with the command; otherwise, #all is false.

1740 #propertylist[] is an array of mandatory non-key property names.

1741 **6.11.2.2.3.2 Pseudo Code**

```
1742 if (false != #all) { #propertylist[] = NULL; }
1743 &smShowAssociationInstances ( "CIM_LogicalIdentity", $instance.getObjectPath(),
1744     #propertylist[] );
1745 &smEnd;
```

1746 **6.11.2.3 Show a Single Instance – Both References**

1747 This command form is for the show verb applied to a single instance. This command form corresponds to
1748 a show command issued against CIM_LogicalIdentity where both references are specified and therefore
1749 the desired instance is unambiguously identified.

1750 **6.11.2.3.1 Command Form**

```
1751 show <CIM_LogicalIdentity single instance>
```

1752 **6.11.2.3.2 CIM Requirements**

1753 See CIM_LogicalIdentity in the “CIM Elements” section of the [Boot Control Profile](#) for the list of mandatory
1754 properties.

1755 **6.11.2.3.3 Behavior Requirements**1756 **6.11.2.3.3.1 Preconditions**

1757 \$instanceA represents the referenced instance of CIM_BootSourceSetting through CIM_LogicalIdentity
1758 association.

1759 \$instanceB represents the other instance of CIM_BootConfigSetting which is referenced by
1760 CIM_LogicalIdentity.

1761 **6.11.2.3.3.2 Pseudo Code**

```
1762 &smShowAssociationInstance ( "CIM_LogicalIdentity", $instanceA.getObjectPath(),
1763     $instanceB.getObjectPath() );
1764 &smEnd;
```

1765 **6.12 CIM_OrderedComponent**

1766 The cd and help verbs shall be supported as described in [DSP0216](#).

1767 Table 12 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
1768 class, and when appropriate, a cross-reference to the section detailing the mapping for the verb and
1769 target. Table 12 is for informational purposes only; in case of a conflict between Table 12 and
1770 requirements detailed in the following sections, the text detailed in the following sections supersedes the
1771 information in Table 12.

1772

Table 12 – Command Verb Requirements for CIM_OrderedComponent

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.12.2.
start	Not supported	
stop	Not supported	

1773 No mappings are defined for the following verbs for the specified target: create, delete, dump, exit,
1774 load, reset, start, and stop.

1775 6.12.1 Ordering of Results

1776 When results are returned for multiple instances of CIM_OrderedComponent, implementations shall
1777 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1778 • Results for CIM_OrderedComponent are unordered; therefore, no algorithm is defined.

1779 6.12.2 Show

1780 This section describes how to implement the `show` verb when applied to an instance of
1781 CIM_OrderedComponent. Implementations shall support the use of the `show` verb with
1782 CIM_OrderedComponent.

1783 6.12.2.1 Show Multiple Instances – CIM_BootConfigSetting Reference

1784 This command form is used when the `show` verb applies to multiple instances. This command form
1785 corresponds to a `show` command issued against instances of CIM_OrderedComponent where only one
1786 reference is specified and the reference is to the instance of CIM_BootConfigSetting.

1787 6.12.3 Command Form

1788 `show <CIM_OrderedComponent multiple instances>`

1789 6.12.3.1.1 CIM Requirements

1790 See CIM_OrderedComponent in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
1791 mandatory properties.

1792 6.12.3.1.2 Behavior Requirements

1793 6.12.3.1.2.1 Preconditions

1794 `$instance` represents the instance of CIM_BootConfigSetting, which is referenced by
1795 CIM_OrderedComponent.

1796 `#all` is true, if the “-all” option was specified with the command; otherwise, `#all` is false.

1797 `#propertylist[]` is an array of mandatory non-key property names.

1798 6.12.3.1.2.2 Pseudo Code

```
1799 if (false != #all) { #propertylist[] = NULL; }  
1800 &smShowAssociationInstances ( "CIM_OrderedComponent", $instance.getObjectPath(),  
1801     #propertylist[] );  
1802 &smEnd;
```

1803 6.12.3.2 Show a Single Instance – CIM_BootSourceSetting Reference

1804 This command form is used when the `show` verb applies to a single instance. This command form
1805 corresponds to a `show` command issued against instances of `CIM_OrderedComponent` where only one
1806 reference is specified and the reference is to the instance of `CIM_BootSourceSetting`. An instance of
1807 `CIM_BootSourceSetting` is referenced by exactly one instance of `CIM_BootConfigSetting`. Therefore, a
1808 single instance is returned.

1809 6.12.3.2.1 Command Form

```
1810 show <CIM_OrderedComponent single instance>
```

1811 6.12.3.2.2 CIM Requirements

1812 See `CIM_OrderedComponent` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
1813 mandatory properties.

1814 6.12.3.2.3 Behavior Requirements**1815 6.12.3.2.3.1 Preconditions**

1816 `$instance` represents the instance of `CIM_BootSourceSetting`, which is referenced by
1817 `CIM_OrderedComponent`.

1818 `#all` is true, if the “-all” option was specified with the command; otherwise, `#all` is false.

1819 `#propertylist[]` is an array of mandatory non-key property names.

1820 6.12.3.2.3.2 Pseudo Code

```
1821 if (false != #all) { #propertylist[] = NULL; }  
1822 &smShowAssociationInstances ( "CIM_OrderedComponent", $instance.getObjectPath(),  
1823     #propertylist[] );  
1824 &smEnd;
```

1825 6.12.3.3 Show a Single Instance – Both References

1826 This command form is for the `show` verb applied to a single instance. This command form corresponds to
1827 a `show` command issued against `CIM_OrderedComponent` where both references are specified and
1828 therefore the desired instance is unambiguously identified.

1829 6.12.3.3.1 Command Form

```
1830 show <CIM_OrderedComponent single instance>
```

1831 6.12.3.3.2 CIM Requirements

1832 See `CIM_OrderedComponent` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
1833 mandatory properties.

1834 **6.12.3.3.3 Behavior Requirements**

1835 **6.12.3.3.3.1 Preconditions**

1836 \$instanceA represents the referenced instance of CIM_BootConfigSetting through
 1837 CIM_OrderedComponent association.

1838 \$instanceB represents the other instance of CIM_BootSourceSetting which is referenced by
 1839 CIM_OrderedComponent.

1840 #all is true, if the "-all" option was specified with the command; otherwise, #all is false.

1841 #propertylist[] is an array of mandatory non-key property names.

1842 **6.12.3.3.3.2 Pseudo Code**

```
1843 if (false != #all) { #propertylist[] = NULL; }
1844 &smShowAssociationInstance ( "CIM_OrderedComponent", $instanceA.getObjectPath(),
1845     $instanceB.getObjectPath(), #propertylist[] );
1846 &smEnd;
```

1847 **6.13 CIM_ServiceAffectsElement**

1848 The cd and help verbs shall be supported as described in [DSP0216](#).

1849 Table 13 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
 1850 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
 1851 target. Table 13 is for informational purposes only; in case of a conflict between Table 13 and
 1852 requirements detailed in the following sections, the text detailed in the following sections supersedes the
 1853 information in Table 13.

1854 **Table 13 – Command Verb Requirements for CIM_ServiceAffectsElement**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.13.2.
Start	Not supported	
Stop	Not supported	

1855 No mappings are defined for the following verbs for the specified target: create, delete, dump, exit,
 1856 load, reset, set, start, and stop.

1857 **6.13.1 Ordering of Results**

1858 When results are returned for multiple instances of CIM_ServiceAffectsElement, implementations shall
 1859 utilize the following algorithm to produce the natural (that is, default) ordering.

- 1860 • Results for CIM_ServiceAffectsElement are unordered; therefore, no algorithm is defined.

1861 6.13.2 Show

1862 This section describes how to implement the `show` verb when applied to an instance of
1863 `CIM_ServiceAffectsElement`. Implementations shall support the use of the `show` verb with
1864 `CIM_ServiceAffectsElement`.

1865 6.13.2.1 Show Multiple Instances – CIM_BootService Reference

1866 This command form is used when the `show` verb applies to multiple instances. This command form
1867 corresponds to a `show` command issued against instances of `CIM_ServiceAffectsElement` where only
1868 one reference is specified and the reference is to an instance of `CIM_BootService`.

1869 6.13.2.1.1 Command Form

```
1870 show <CIM_ServiceAffectsElement multiple instances>
```

1871 6.13.2.1.2 CIM Requirements

1872 See `CIM_ServiceAffectsElement` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
1873 mandatory properties.

1874 6.13.2.1.3 Behavior Requirements**1875 6.13.2.1.3.1 Preconditions**

1876 `$instance` represents the instance of `CIM_BootService`, which is referenced by
1877 `CIM_ServiceAffectsElement`.

1878 6.13.2.1.3.2 Pseudo Code

```
1879 &smShowAssociationInstances ( "CIM_ServiceAffectsElement",  
1880     $instance.GetObjectPath() );  
1881 &smEnd;
```

1882 6.13.2.2 Show a Single Instance – CIM_ComputerSystem Reference

1883 This command form is used when the `show` verb applies to multiple instances. This command form
1884 corresponds to a `show` command issued against instances of `CIM_ServiceAffectsElement` where only
1885 one reference is specified and the reference is to an instance of `CIM_ComputerSystem`. An instance of
1886 `CIM_BootService` is referenced by exactly one instance of `CIM_ComputerSystem`. Therefore, a single
1887 instance is returned.

1888 6.13.2.2.1 Command Form

```
1889 show <CIM_ServiceAffectsElement single instance>
```

1890 6.13.2.2.2 CIM Requirements

1891 See `CIM_ServiceAffectsElement` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
1892 mandatory properties.

1893 6.13.2.2.3 Behavior Requirements**1894 6.13.2.2.3.1 Preconditions**

1895 `$instance` represents the instance of `CIM_ComputerSystem`, which is referenced by
1896 `CIM_ServiceAffectsElement`.

1897 6.13.2.2.3.2 Pseudo Code

```
1898 &smShowAssociationInstance ( "CIM_ServiceAffectsElement", $instance.getObjectPath() );  
1899 &smEnd;
```

1900 6.13.2.3 Show a Single Instance – Both References

1901 This command form is used when the `show` verb applies to a single instance. This command form
1902 corresponds to a `show` verb issued against an instance of `CIM_ServiceAffectsElement` where both
1903 references are specified and therefore the desired instance is unambiguously identified.

1904 6.13.2.3.1 Command Form

```
1905 show <CIM_ServiceAffectsElement single instance>
```

1906 6.13.2.3.2 CIM Requirements

1907 See `CIM_ServiceAffectsElement` in the “CIM Elements” section of the [Boot Control Profile](#) for the list of
1908 mandatory properties.

1909 6.13.2.3.3 Behavior Requirements**1910 6.13.2.3.3.1 Preconditions**

1911 `$instanceA` represents the referenced instance of `CIM_BootService` through
1912 `CIM_ServiceAffectsElement` association.

1913 `$instanceB` represents the other instance of `CIM_ComputerSystem` which is referenced by
1914 `CIM_ServiceAffectsElement`.

1915 6.13.2.3.3.2 Pseudo Code

```
1916 &smShowAssociationInstance ( "CIM_ServiceAffectsElement", $instanceA.getObjectPath(),  
1917     $instanceB.getObjectPath() );  
1918 &smEnd;
```

1919

1920
1921
1922
1923
1924

ANNEX A
(informative)

Change Log

Version	Date	Author	Description
1.0.0	2009-06-04		DMTF Standard Release

1925