



1
2
3
4

Document Number: DSP0821

Date: 2009-07-14

Version: 1.0.0

5
6

SSH Service Profile SM CLP Command Mapping 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27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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 5.1 IShowTCPEndpoint..... 10

45 6 Mappings..... 10

46 6.1 CIM_BindsTo 10

47 6.2 CIM_ElementCapabilities 13

48 6.3 CIM_ElementSettingData 16

49 6.4 CIM_HostedAccessPoint 21

50 6.5 CIM_HostedService 24

51 6.6 CIM_ProvidesEndpoint 26

52 6.7 CIM_ProtocolService 28

53 6.8 CIM_ServiceAccessBySAP 33

54 6.9 CIM_SSHTCapabilities 35

55 6.10 CIM_SSHProtocolEndpoint 37

56 6.11 CIM_SSHSettingData 42

57 6.12 CIM_TCPProtocolEndpoint..... 45

58 ANNEX A (informative) Change Log 53

59

60 Tables

61 Table 1 – Local Recipes..... 9

62 Table 2 – Command Verb Requirements for CIM_BindsTo 10

63 Table 3 – Command Verb Requirements for CIM_ElementCapabilities 14

64 Table 4 – Command Verb Requirements for CIM_ElementSettingData 16

65 Table 5 – Command Verb Requirements for CIM_HostedAccessPoint 22

66 Table 6 – Command Verb Requirements for CIM_HostedService 24

67 Table 7 – Command Verb Requirements for CIM_ProvidesEndpoint 26

68 Table 8 – Command Verb Requirements for CIM_ProtocolService 28

69 Table 9 – Command Verb Requirements for CIM_ServiceAccessBySAP 33

70 Table 10 – Command Verb Requirements for CIM_SSHTCapabilities 35

71 Table 11 – Command Verb Requirements for CIM_SSHProtocolEndpoint 37

72 Table 12 – Command Verb Requirements for CIM_SSHSettingData 42

73 Table 13 – Command Verb Requirements for CIM_TCPProtocolEndpoint..... 45

74

76

Foreword

77 The *SSH Service Profile SM CLP Command Mapping Specification* (DSP0821) 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 SNIA
81 [SMI-S 1.1.0](#), section 7.6.

82 **Acknowledgements**

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

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

92

93

Introduction

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

98 The target audience for this specification is implementers of the SM CLP support for the [SSH Service](#)
99 [Profile](#).

100
101

SSH Service Profile SM CLP Command Mapping Specification

1 Scope

103 This specification contains the requirements for an implementation of the SM CLP to provide access to,
104 and implement the behaviors of, the [SSH Service Profile](#).

2 Normative References

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

2.1 Approved References

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

112 DMTF DSP1006, *SMASH Collections Profile 1.0*,
113 http://www.dmtf.org/standards/published_documents/DSP1006_1.0.pdf

114 DMTF DSP1017, *SSH Service Profile 1.0*,
115 http://www.dmtf.org/standards/published_documents/DSP1017_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

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>

3 Terms and Definitions

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

3.1

can

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

3.2

cannot

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

3.3

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 Common Information Model
- 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 • smStartRSC()
- 191 • smStopRSC()
- 192 • smResetRSC()
- 193 • smShowInstance()
- 194 • smShowInstances()
- 195 • smSetInstance()
- 196 • smShowAssociationInstances()
- 197 • smShowAssociationInstance()
- 198 • smDeleteInstance
- 199 • smMakeCommandStatus
- 200 • smNewInstance

201 For convenience, Table 1 lists each recipe defined in this mapping which is used for more than one verb
 202 or class mapping.

203 **Table 1 – Local Recipes**

| Recipe Name | Description | Definition |
|------------------|---|------------|
| IShowTCPEndpoint | Show an instance of CIM_TCPProtocolEndpoint | See 5.1. |

204 The following sections detail Local Recipes defined for use in this mapping.

205 5.1 IShowTCPEndpoint

206 5.1.1 Description

207 Reusable recipe for displaying an instance of CIM_TCPProtocolEndpoint. A recipe is defined for re-use
208 by the `show` and `create` verbs applied to CIM_TCPProtocolEndpoint.

209 5.1.2 Preconditions

210 `$endpoint` contains the instance of CIM_TCPProtocolEndpoint to display.

211 `#all` indicates whether the “-all” option was specified.

212 5.1.3 Pseudo Code

```
213 sub lShowTCPEndpoint($endpoint, #all)
214 {
215     #propertylist[] = NULL;
216     //if we're not displaying all of the properties, provide a list
217     if (false == #all)
218     {
219         #propertylist[] = { //all mandatory non-key properties };
220     }
221     &smShowInstance ( $endpoint.GetObjectPath(), #propertyList[] );
222     &smEnd;
223 } //lShowTCPEndpoint()
```

224 6 Mappings

225 The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in
226 the [SSH Service Profile](#). Requirements specified here related to support for a CLP verb for a particular
227 class are solely within the context of this profile.

228 6.1 CIM_BindsTo

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

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

235 **Table 2 – Command Verb Requirements for CIM_BindsTo**

| Command Verb | Requirement | Comments |
|--------------|---------------|----------|
| create | Not supported | |
| delete | Not supported | |
| dump | Not supported | |
| load | Not supported | |
| reset | Not supported | |

| Command Verb | Requirement | Comments |
|--------------|---------------|------------|
| set | Not supported | |
| show | Shall | See 6.1.2. |
| start | Not supported | |
| stop | Not supported | |

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

238 6.1.1 Ordering of Results

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

- 241 • Results for `CIM_BindsTo` are unordered; therefore, no algorithm is defined.

242 6.1.2 Show

243 This section describes how to implement the `show` verb when applied to an instance of `CIM_BindsTo`.
 244 Implementations shall support the use of the `show` verb with `CIM_BindsTo`.

245 The `show` command is used to display information about the `CIM_BindsTo` instance or instances.

246 6.1.2.1 Show Multiple Instances – CIM_IPProtocolEndpoint

247 This command form is for the `show` verb applied to multiple instances. This command form corresponds
 248 to a `show` command issued against `CIM_BindsTo` where only one reference is specified and the
 249 reference is to an instance of `CIM_IPProtocolEndpoint`.

250 6.1.2.1.1 Command Form

```
251 show <CIM_BindsTo multiple instances>
```

252 6.1.2.1.2 CIM Requirements

253 See `CIM_BindsTo` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of mandatory
 254 properties.

255 6.1.2.1.3 Behavior Requirements

256 6.1.2.1.3.1 Preconditions

257 `instance` contains the instance of `CIM_IPProtocolEndpoint` which is referenced by `CIM_BindsTo`.

258 6.1.2.1.3.2 Pseudo Code

```
259 &smShowAssociationInstances ( "CIM_BindsTo", $instance.getObjectPath() );  
260 &smEnd;
```

261 6.1.2.2 Show Multiple Instances – CIM_TCPProtocolEndpoint

262 This command form is for the `show` verb applied to multiple instances. This command form corresponds
 263 to a `show` command issued against `CIM_BindsTo` where only one reference is specified and the
 264 reference is to an instance of `CIM_TCPProtocolEndpoint`.

265 6.1.2.2.1 Command Form

```
266 show <CIM_BindsTo multiple instances>
```

267 6.1.2.2.2 CIM Requirements

268 See CIM_BindsTo in the “CIM Elements” section of the [SSH Service Profile](#) for the list of mandatory
269 properties.

270 6.1.2.2.3 Behavior Requirements**271 6.1.2.2.3.1 Preconditions**

272 \$instance contains the instance of CIM_TCPProtocolEndpoint which is referenced by CIM_BindsTo.

273 6.1.2.2.3.2 Pseudo Code

```
274 &smShowAssociationInstances ( "CIM_BindsTo", $instance.getObjectPath() );  
275 &smEnd;
```

276 6.1.2.3 Show a Single Instance – CIM_SSHProtocolEndpoint Reference

277 This command form is for the `show` verb applied to a single instance. This command form corresponds to
278 a `show` command issued against CIM_BindsTo where the reference specified is to an instance of
279 CIM_SSHProtocolEndpoint. A single instance of CIM_TCPProtocolEndpoint is associated with each
280 instance of CIM_SSHProtocolEndpoint. Therefore, a single instance will be returned.

281 6.1.2.3.1 Command Form

```
282 show <CIM_BindsTo single instance>
```

283 6.1.2.3.2 CIM Requirements

284 See CIM_BindsTo in the “CIM Elements” section of the [SSH Service Profile](#) for the list of mandatory
285 properties.

286 6.1.2.3.3 Behavior Requirements**287 6.1.2.3.3.1 Preconditions**

288 \$instance contains the instance of CIM_SSHProtocolEndpoint which is referenced by CIM_BindsTo.

289 6.1.2.3.3.2 Pseudo Code

```
290 &smShowAssociationInstances ( "CIM_BindsTo", $instance.getObjectPath() );  
291 &smEnd;
```

292 6.1.2.4 Show a Single Instance – Both References A

293 This command form is for the `show` verb applied to a single instance. This command form corresponds to
294 a `show` command issued against CIM_BindsTo where a reference to CIM_SSHProtocolEndpoint and a
295 reference to CIM_TCPProtocolEndpoint are specified and therefore the desired instance is
296 unambiguously identified.

297 6.1.2.4.1 Command Form

```
298 show <CIM_BindsTo single instance>
```

299 6.1.2.4.2 CIM Requirements

300 See CIM_BindsTo in the “CIM Elements” section of the [SSH Service Profile](#) for the list of mandatory
301 properties.

302 6.1.2.4.3 Behavior Requirements

303 6.1.2.4.3.1 Preconditions

304 \$instanceA contains the instance of CIM_SSHProtocolEndpoint which is referenced by CIM_BindsTo.

305 \$instanceB contains the instance of CIM_TCPProtocolEndpoint which is referenced by CIM_BindsTo.

306 6.1.2.4.3.2 Pseudo Code

```
307 &smShowAssociationInstance ( "CIM_BindsTo", $instanceA.getObjectPath(),
308     $instanceB.getObjectPath() );
309 &smEnd;
```

310 6.1.2.5 Show a Single Instance – Both References B

311 This command form is for the show verb applied to a single instance. This command form corresponds to
312 a show command issued against CIM_BindsTo where a reference to CIM_IPProtocolEndpoint and a
313 reference to CIM_TCPProtocolEndpoint are specified and therefore the desired instance is
314 unambiguously identified.

315 6.1.2.5.1 Command Form

```
316 show <CIM_BindsTo single instance>
```

317 6.1.2.5.2 CIM Requirements

318 See CIM_BindsTo in the “CIM Elements” section of the [SSH Service Profile](#) for the list of mandatory
319 properties.

320 6.1.2.5.3 Behavior Requirements

321 6.1.2.5.3.1 Preconditions

322 \$instanceA contains the instance of CIM_IPProtocolEndpoint which is referenced by CIM_BindsTo.

323 \$instanceB contains the instance of CIM_TCPProtocolEndpoint which is referenced by CIM_BindsTo.

324 6.1.2.5.3.2 Pseudo Code

```
325 &smShowAssociationInstance ( "CIM_BindsTo", $instanceA.getObjectPath(),
326     $instanceB.getObjectPath() );
327 &smEnd;
```

328 6.2 CIM_ElementCapabilities

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

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

335

Table 3 – 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 6.2.2. |
| start | Not supported | |
| stop | Not supported | |

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

338 6.2.1 Ordering of Results

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

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

342 6.2.2 Show

343 This section describes how to implement the `show` verb when applied to an instance of
 344 `CIM_ElementCapabilities`. Implementations shall support the use of the `show` verb with
 345 `CIM_ElementCapabilities`.

346 The `show` command is used to display information about the `CIM_ElementCapabilities` instance or
 347 instances.

348 6.2.2.1 Show a Single Instance – CIM_SSHCapabilities Reference

349 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 350 a `show` command issued against `CIM_ElementCapabilities` where the reference specified is to an
 351 instance of `CIM_SSHCapabilities`. A single instance of `CIM_ProtocolService` is associated with each
 352 instance of a `CIM_SSHCapabilities`. Therefore, a single instance will be returned.

353 6.2.2.1.1 Command Form

```
354 show <CIM_ElementCapabilities single instance>
```

355 6.2.2.1.2 CIM Requirements

356 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
 357 mandatory properties.

358 6.2.2.1.3 Behavior Requirements

359 6.2.2.1.3.1 Preconditions

360 \$instance contains the instance of CIM_SSHCapabilities which is referenced by
361 CIM_ElementCapabilities.

362 6.2.2.1.3.2 Pseudo Code

```
363 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );
364 &smEnd;
```

365 6.2.2.2 Show a Single Instance – CIM_ProtocolService Reference

366 This command form is for the `show` verb applied to a single instance. This command form corresponds to
367 a `show` command issued against `CIM_ElementCapabilities` where the reference specified is to an
368 instance of `CIM_ProtocolService`. A single instance of `CIM_SSHCapabilities` is associated with each
369 instance of `CIM_ProtocolService`. Therefore, a single instance will be returned.

370 6.2.2.2.1 Command Form

```
371 show <CIM_ElementCapabilities single instance>
```

372 6.2.2.2.2 CIM Requirements

373 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
374 mandatory properties.

375 6.2.2.2.3 Behavior Requirements

376 6.2.2.2.3.1 Preconditions

377 \$instance contains the instance of `CIM_ProtocolService` which is referenced by
378 `CIM_ElementCapabilities`.

379 6.2.2.2.3.2 Pseudo Code

```
380 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );
381 &smEnd;
```

382 6.2.2.3 Show a Single Instance – Both References

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

386 6.2.2.3.1 Command Form

```
387 show <CIM_ElementCapabilities single instance>
```

388 6.2.2.3.2 CIM Requirements

389 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
390 mandatory properties.

391 **6.2.2.3.3 Behavior Requirements**392 **6.2.2.3.3.1 Preconditions**

393 \$instanceA contains the instance of CIM_SSHCapabilities which is referenced by
394 CIM_ElementCapabilities.

395 \$instanceB contains the instance of CIM_ProtocolService which is referenced by
396 CIM_ElementCapabilities.

397 **6.2.2.3.3.2 Pseudo Code**

```
398 &smShowAssociationInstance ( "CIM_ElementCapabilities", $instanceA.getObjectPath(),
399     $instanceB.getObjectPath() );
400 &smEnd;
```

401 **6.3 CIM_ElementSettingData**

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

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

408 **Table 4 – 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 6.3.2. |
| show | Shall | See 6.3.3. |
| start | Not supported | |
| stop | Not supported | |

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

411 **6.3.1 Ordering of Results**

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

- 414
- Results for CIM_ElementSettingData are unordered; therefore, no algorithm is defined.

415 **6.3.2 Set**

416 This section describes how to implement the `set` verb when it is applied to an instance of
 417 `CIM_ElementSettingData`. Implementations may support the use of the `set` verb with
 418 `CIM_ElementSettingData`.

419 The `set` verb is used to modify properties of the `CIM_ElementSettingData` instance.

420 **6.3.2.1 Set of IsNext**

421 The `IsNext` property is the only property of `CIM_ElementSettingData` which can be modified directly via
 422 the `set` verb.

423 **6.3.2.1.1 Command Form**

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

425 **6.3.2.1.2 CIM Requirements**

426 See `CIM_ElementSettingData` in the “CIM Elements” section of the [SSH Service Profile](#) for the
 427 `CIM_ElementSettingData.IsNext` property.

428 **6.3.2.1.3 Behavior Requirements**

```
429 $instance=<CIM_ElementSettingData single instance>
430 #propertyName[] = { "IsNext" };
431 #propertyValues[] = {<propertyvalue>};
432 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
433 &smEnd;
```

434 **6.3.3 Show**

435 This section describes how to implement the `show` verb when applied to an instance of
 436 `CIM_ElementSettingData`. Implementations shall support the use of the `show` verb with
 437 `CIM_ElementSettingData`.

438 The `show` command is used to display information about the `CIM_ElementSettingData` instance or
 439 instances.

440 **6.3.3.1 Show Multiple Instances – CIM_SSHSettingData and CIM_SSHProtocolEndpoint**

441 This command form corresponds to a `show` command issued against `CIM_ElementSettingData` where
 442 the reference specified is to an instance of `CIM_SSHSettingData`. Note that when an instance of
 443 `CIM_SSHSettingData` is associated with an instance of `CIM_SSHProtocolEndpoint`, the `IsCurrent`
 444 property is the mandatory property.

445 **6.3.3.1.1 Command Form**

```
446 show <CIM_ElementSettingData multiple instances>
```

447 **6.3.3.1.2 CIM Requirements**

448 See `CIM_ElementSettingData` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
 449 mandatory properties.

450 6.3.3.1.3 Behavior Requirements

451 6.3.3.1.3.1 Preconditions

452 \$instance contains the instance of CIM_SSHSettingData which is referenced by
453 CIM_ElementSettingData.

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

455 6.3.3.1.3.2 Pseudo Code

```
456 #propertylist = NULL;  
457 if (false == #all)  
458     {  
459         #propertylist = { "IsCurrent" };  
460     }  
461 &smShowAssociationInstances ( "CIM_ElementSettingData", $instance.getObjectPath(),  
462     #propertylist[] );  
463 &smEnd;
```

464 6.3.3.2 Show Multiple Instances – CIM_SSHProtocolEndpoint Reference

465 This command form corresponds to a show command issued against CIM_ElementSettingData where
466 the reference specified is to an instance of CIM_SSHProtocolEndpoint. Note that when an instance of
467 CIM_SSHSettingData is associated with an instance of CIM_SSHProtocolEndpoint, the IsCurrent
468 property is the mandatory property.

469 6.3.3.2.1 Command Form

```
470 show <CIM_ElementSettingData multiple instances>
```

471 6.3.3.2.2 CIM Requirements

472 See CIM_ElementSettingData in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
473 mandatory properties.

474 6.3.3.2.3 Behavior Requirements

475 6.3.3.2.3.1 Preconditions

476 \$instance contains the instance of CIM_SSHProtocolEndpoint which is referenced by
477 CIM_ElementSettingData.

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

479 6.3.3.2.3.2 Pseudo Code

```
480 #propertylist = NULL;  
481 if (false == #all)  
482     {  
483         #propertylist = { "IsCurrent" };  
484     }  
485 &smShowAssociationInstances ( "CIM_ElementSettingData", $instance.getObjectPath(),  
486     #propertylist[] );  
487 &smEnd;
```

488 6.3.3.3 Show a Single Instance – CIM_SSHSettingData and CIM_SSHProtocolEndpoint

489 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 490 a `show` command issued against `CIM_ElementSettingData` where both references are specified and
 491 therefore the desired instance is unambiguously identified.

492 6.3.3.3.1 Command Form

```
493 show <CIM_ElementSettingData single instance>
```

494 6.3.3.3.2 CIM Requirements

495 See `CIM_ElementSettingData` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
 496 mandatory properties.

497 6.3.3.3.3 Behavior Requirements

498 6.3.3.3.3.1 Preconditions

499 `$instanceA` contains the instance of `CIM_SSHSettingData` which is referenced by
 500 `CIM_ElementSettingData`.

501 `$instanceB` contains the instance of `CIM_SSHProtocolEndpoint` which is referenced by
 502 `CIM_ElementSettingData`.

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

504 6.3.3.3.3.2 Pseudo Code

```
505 #propertylist = NULL;
506 if (false == #all)
507     {
508         #propertylist = { "IsCurrent" };
509     }
510 &smShowAssociationInstance ( "CIM_ElementSettingData", $instanceA.getObjectPath(),
511     $instanceB.getObjectPath(), #propertylist[] );
512 &smEnd;
```

513 6.3.3.4 Show Multiple Instances – CIM_SSHSettingData and CIM_ProtocolService

514 This command form corresponds to a `show` command issued against `CIM_ElementSettingData` where
 515 the reference specified is to an instance of `CIM_SSHSettingData`. Note that when an instance of
 516 `CIM_SSHSettingData` is associated with an instance of `CIM_ProtocolService`, the `IsNext` and `IsDefault`
 517 properties are mandatory.

518 6.3.3.4.1 Command Form

```
519 show <CIM_ElementSettingData multiple instances>
```

520 6.3.3.4.2 CIM Requirements

521 See `CIM_ElementSettingData` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
 522 mandatory properties.

523 6.3.3.4.3 Behavior Requirements

524 6.3.3.4.3.1 Preconditions

525 \$instance contains the instance of CIM_SSHSettingData which is referenced by
526 CIM_ElementSettingData.

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

528 6.3.3.4.3.2 Pseudo Code

```
529 #propertylist[] = NULL;  
530 if (false == #all)  
531     {  
532         #propertylist = { "IsNext", "IsDefault" };  
533     }  
534 &smShowAssociationInstances ( "CIM_ElementSettingData", $instance.getObjectPath(),  
535     #propertylist[] );  
536 &smEnd;
```

537 6.3.3.5 Show Multiple Instances – CIM_ProtocolService Reference

538 This command form corresponds to a show command issued against CIM_ElementSettingData where
539 the reference specified is to an instance of CIM_ProtocolService. Note that when an instance of
540 CIM_SSHSettingData is associated with an instance of CIM_ProtocolService, the IsNext and IsDefault
541 properties are mandatory.

542 6.3.3.5.1 Command Form

```
543 show <CIM_ElementSettingData multiple instances>
```

544 6.3.3.5.2 CIM Requirements

545 See CIM_ElementSettingData in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
546 mandatory properties.

547 6.3.3.5.3 Behavior Requirements

548 6.3.3.5.3.1 Preconditions

549 \$instance contains the instance of CIM_ProtocolService which is referenced by
550 CIM_ElementSettingData.

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

552 6.3.3.5.3.2 Pseudo Code

```
553 #propertylist[] = NULL;  
554 if (false == #all)  
555     {  
556         #propertylist = { "IsNext", "IsDefault" };  
557     }  
558 &smShowAssociationInstances ( "CIM_ElementSettingData", $instance.getObjectPath(),  
559     #propertylist[] );  
560 &smEnd;
```

561 **6.3.3.6 Show a Single Instance – SettingData and ProtocolService**

562 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 563 a `show` command issued against `CIM_ElementSettingData` where both references are specified and
 564 therefore the desired instance is unambiguously identified.

565 **6.3.3.6.1 Command Form**

```
566 show <CIM_ElementSettingData single instance>
```

567 **6.3.3.6.2 CIM Requirements**

568 See `CIM_ElementSettingData` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
 569 mandatory properties.

570 **6.3.3.6.3 Behavior Requirements**

571 **6.3.3.6.3.1 Preconditions**

572 `$instanceA` contains the instance of `CIM_SSHSettingData` which is referenced by
 573 `CIM_ElementSettingData`.

574 `$instanceB` contains the instance of `CIM_ProtocolService` which is referenced by
 575 `CIM_ElementSettingData`.

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

577 **6.3.3.6.3.2 Pseudo Code**

```
578 #propertylist[] = NULL;
579 if (false == #all)
580 {
581     #propertylist = { "IsNext", "IsDefault" };
582 }
583 &smShowAssociationInstance ( "CIM_ElementSettingData", $instanceA.getObjectPath(),
584     $instanceB.getObjectPath(), #propertylist[] );
585 &smEnd;
```

586 **6.4 CIM_HostedAccessPoint**

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

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

593

Table 5 – Command Verb Requirements for CIM_HostedAccessPoint

| 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 6.4.2. |
| start | Not supported | |
| stop | Not supported | |

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

596 6.4.1 Ordering of Results

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

- 599 • Results for `CIM_HostedAccessPoint` are unordered; therefore, no algorithm is defined.

600 6.4.2 Show

601 This section describes how to implement the `show` verb when applied to an instance of
 602 `CIM_HostedAccessPoint`. Implementations shall support the use of the `show` verb with
 603 `CIM_HostedAccessPoint`.

604 The `show` command is used to display information about the `CIM_HostedAccessPoint` instance or
 605 instances.

606 6.4.2.1 Show Multiple Instances

607 This command form is for the `show` verb applied to multiple instances. This command form corresponds
 608 to a `show` command issued against `CIM_HostedAccessPoint` where only one reference is specified and
 609 the reference is to an instance of `CIM_ComputerSystem`.

610 6.4.2.1.1 Command Form

```
611 show <CIM_HostedAccessPoint multiple instances>
```

612 6.4.2.1.2 CIM Requirements

613 See `CIM_HostedAccessPoint` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
 614 mandatory properties.

615 6.4.2.1.3 Behavior Requirements

616 6.4.2.1.3.1 Preconditions

617 `$instance` contains the instance of `CIM_ComputerSystem` which is referenced by
 618 `CIM_HostedAccessPoint`.

619 **6.4.2.1.3.2 Pseudo Code**

```
620 &smShowAssociationInstances ( "CIM_HostedAccessPoint", $instance.getObjectPath() );
621 &smEnd;
```

622 **6.4.2.2 Show a Single Instance – CIM_TCPProtocolEndpoint or CIM_SSHProtocolEndpoint Reference**

624 This command form is for the `show` verb applied to a single instance. This command form corresponds to
625 a `show` command issued against `CIM_HostedAccessPoint` where the reference specified is to an
626 instance of `CIM_TCPProtocolEndpoint` or `CIM_SSHProtocolEndpoint`. A single instance will be returned.

627 **6.4.2.2.1 Command Form**

```
628 show <CIM_HostedAccessPoint single instance>
```

629 **6.4.2.2.2 CIM Requirements**

630 See `CIM_HostedAccessPoint` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
631 mandatory properties.

632 **6.4.2.2.3 Behavior Requirements**

633 **6.4.2.2.3.1 Preconditions**

634 `$instance` contains the instance of `CIM_TCPProtocolEndpoint` or `CIM_SSHProtocolEndpoint` which is
635 referenced by `CIM_HostedAccessPoint`.

636 **6.4.2.2.3.2 Pseudo Code**

```
637 &smShowAssociationInstances ( "CIM_HostedAccessPoint", $instance.getObjectPath() );
638 &smEnd;
```

639 **6.4.2.3 Show a Single Instance – Both References**

640 This command form is for the `show` verb applied to a single instance. This command form corresponds to
641 a `show` command issued against `CIM_HostedAccessPoint` where both references are specified and
642 therefore the desired instance is unambiguously identified.

643 **6.4.2.3.1 Command Form**

```
644 show <CIM_HostedAccessPoint single instance>
```

645 **6.4.2.3.2 CIM Requirements**

646 See `CIM_HostedAccessPoint` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
647 mandatory properties.

648 **6.4.2.3.3 Behavior Requirements**

649 **6.4.2.3.3.1 Preconditions**

650 `$instanceA` contains the instance of `CIM_ComputerSystem` which is referenced by
651 `CIM_HostedAccessPoint`.

652 `$instanceB` contains the instance of `CIM_SSHProtocolEndpoint` or `CIM_TCPProtocolEndpoint` which is
653 referenced by `CIM_HostedAccessPoint`.

654 **6.4.2.3.3.2 Pseudo Code**

```
655 &smShowAssociationInstance ( "CIM_HostedAccessPoint", $instanceA.getObjectPath(),
656     $instanceB.getObjectPath() );
657 &smEnd;
```

658 **6.5 CIM_HostedService**

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

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

665 **Table 6 – 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 6.5.2. |
| start | Not supported | |
| stop | Not supported | |

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

668 **6.5.1 Ordering of Results**

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

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

672 **6.5.2 Show**

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

675 The `show` command is used to display information about the `CIM_HostedService` instance or instances.

676 **6.5.2.1 Show Multiple Instances**

677 This command form is for the `show` verb applied to multiple instances. This command form corresponds
 678 to a `show` command issued against `CIM_HostedService` where only one reference is specified and the
 679 reference is to an instance of `CIM_ComputerSystem`.

680 **6.5.2.1.1 Command Form**681 `show <CIM_HostedService multiple instances>`682 **6.5.2.1.2 CIM Requirements**683 See CIM_HostedService in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
684 mandatory properties.685 **6.5.2.1.3 Behavior Requirements**686 **6.5.2.1.3.1 Preconditions**

687 \$instance contains the instance of CIM_ComputerSystem which is referenced by CIM_HostedService.

688 **6.5.2.1.3.2 Pseudo Code**689 `&smShowAssociationInstances ("CIM_HostedService", $instance.getObjectPath());`
690 `&smEnd;`691 **6.5.2.2 Show a Single Instance – CIM_ProtocolService Reference**692 This command form is for the `show` verb applied to a single instance. This command form corresponds to
693 a `show` command issued against CIM_HostedService where the reference specified is to an instance of
694 CIM_ProtocolService. An instance of CIM_ProtocolService is referenced by exactly one instance of
695 CIM_HostedService. Therefore, a single instance will be returned.696 **6.5.2.2.1 Command Form**697 `show <CIM_HostedService single instance>`698 **6.5.2.2.2 CIM Requirements**699 See CIM_HostedService in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
700 mandatory properties.701 **6.5.2.2.3 Behavior Requirements**702 **6.5.2.2.3.1 Preconditions**

703 \$instance contains the instance of CIM_ProtocolService which is referenced by CIM_HostedService.

704 **6.5.2.2.3.2 Pseudo Code**705 `&smShowAssociationInstances ("CIM_HostedService", $instance.getObjectPath());`
706 `&smEnd;`707 **6.5.2.3 Show a Single Instance – Both References**708 This command form is for the `show` verb applied to a single instance. This command form corresponds to
709 a `show` command issued against CIM_HostedService where both references are specified and therefore
710 the desired instance is unambiguously identified.711 **6.5.2.3.1 Command Form**712 `show <CIM_HostedService single instance>`

713 **6.5.2.3.2 CIM Requirements**

714 See CIM_HostedService in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
715 mandatory properties.

716 **6.5.2.3.3 Behavior Requirements**717 **6.5.2.3.3.1 Preconditions**

718 \$instanceA contains the instance of CIM_ComputerSystem which is referenced by
719 CIM_HostedService.

720 \$instanceB contains the instance of CIM_ProtocolService which is referenced by CIM_HostedService.

721 **6.5.2.3.3.2 Pseudo Code**

```
722 &smShowAssociationInstance ( "CIM_HostedService", $instanceA.getObjectPath(),
723     $instanceB.getObjectPath() );
724 &smEnd;
```

725 **6.6 CIM_ProvidesEndpoint**

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

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

732 **Table 7 – Command Verb Requirements for CIM_ProvidesEndpoint**

| 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 | |

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

735 **6.6.1 Ordering of Results**

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

- 738 • Results for CIM_ProvidesEndpoint are unordered; therefore, no algorithm is defined.

739 **6.6.2 Show**

740 This section describes how to implement the `show` verb when applied to an instance of
 741 `CIM_ProvidesEndpoint`. Implementations shall support the use of the `show` verb with
 742 `CIM_ProvidesEndpoint`.

743 The `show` command is used to display information about the `CIM_ProvidesEndpoint` instance or
 744 instances.

745 **6.6.2.1 Show a Single Instance – CIM_SSHProtocolEndpoint Reference**

746 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 747 a `show` command issued against `CIM_ProvidesEndpoint` where the reference specified is to an instance
 748 of `CIM_SSHProtocolEndpoint`. A single instance of `CIM_ProtocolService` is associated with each instance
 749 of a `CIM_SSHProtocolEndpoint`. Therefore, a single instance will be returned.

750 **6.6.2.1.1 Command Form**

```
751 show <CIM_ProvidesEndpoint single instance>
```

752 **6.6.2.1.2 CIM Requirements**

753 See `CIM_ProvidesEndpoint` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
 754 mandatory properties.

755 **6.6.2.1.3 Behavior Requirements**

756 **6.6.2.1.3.1 Preconditions**

757 `$instance` contains the instance of `CIM_SSHProtocolEndpoint` which is referenced by
 758 `CIM_ProvidesEndpoint`.

759 **6.6.2.1.3.2 Pseudo Code**

```
760 &smShowAssociationInstances ( "CIM_ProvidesEndpoint", $instance.getObjectPath() );  
761 &smEnd;
```

762 **6.6.2.2 Show Multiple Instances – CIM_ProtocolService Reference**

763 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 764 a `show` command issued against `CIM_ProvidesEndpoint` where the reference specified is to an instance
 765 of `CIM_ProtocolService`. A single instance of `CIM_ProtocolService` is associated with multiple instances
 766 of a `CIM_SSHProtocolEndpoint`. Therefore, multiple instances may be returned.

767 **6.6.2.2.1 Command Form**

```
768 show <CIM_ProvidesEndpoint multiple instances>
```

769 **6.6.2.2.2 CIM Requirements**

770 See `CIM_ProvidesEndpoint` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
 771 mandatory properties.

772 **6.6.2.2.3 Behavior Requirements**

773 **6.6.2.2.3.1 Preconditions**

774 `$instance` contains the instance of `CIM_ProtocolService` which is referenced by
 775 `CIM_ProvidesEndpoint`.

776 **6.6.2.2.3.2 Pseudo Code**

```
777 &smShowAssociationInstances ( "CIM_ProvidesEndpoint", $instance.getObjectPath() );
778 &smEnd;
```

779 **6.6.2.3 Show a Single Instance – Both References**

780 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 781 a `show` command issued against `CIM_ProvidesEndpoint` where both references are specified and
 782 therefore the desired instance is unambiguously identified.

783 **6.6.2.3.1 Command Form**

```
784 show <CIM_ProvidesEndpoint single instance>
```

785 **6.6.2.3.2 CIM Requirements**

786 See `CIM_ProvidesEndpoint` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
 787 mandatory properties.

788 **6.6.2.3.3 Behavior Requirements**789 **6.6.2.3.3.1 Preconditions**

790 `$instanceA` contains the instance of `CIM_SSHProtocolEndpoint` which is referenced by
 791 `CIM_ProvidesEndpoint`.

792 `$instanceB` contains the instance of `CIM_ProtocolService` which is referenced by
 793 `CIM_ProvidesEndpoint`.

794 **6.6.2.3.3.2 Pseudo Code**

```
795 &smShowAssociationInstance ( "CIM_ProvidesEndpoint", $instanceA.getObjectPath(),
796     $instanceB.getObjectPath() );
797 &smEnd;
```

798 **6.7 CIM_ProtocolService**

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

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

805 **Table 8 – Command Verb Requirements for CIM_ProtocolService**

| Command Verb | Requirement | Comments |
|--------------|---------------|------------|
| create | Not supported | |
| delete | Not supported | |
| dump | Not supported | |
| load | Not supported | |
| reset | May | See 6.7.2. |
| set | May | See 6.7.3. |

| Command Verb | Requirement | Comments |
|--------------|-------------|------------|
| show | Shall | See 6.7.4. |
| start | May | See 6.7.5. |
| stop | May | See 6.7.6. |

806 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, and `load`.

807 6.7.1 Ordering of Results

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

- 810 • Results for `CIM_ProtocolService` are unordered; therefore, no algorithm is defined.

811 6.7.2 Reset

812 This section describes how to implement the `reset` verb when applied to an instance of
813 `CIM_ProtocolService`. Implementations may support the use of the `reset` verb with
814 `CIM_ProtocolService`.

815 The `reset` verb is used to initiate a reset of the `CIM_ProtocolService`.

816 6.7.2.1 Reset a Single Instance

817 This command form is for the initiation of a reset action against a single instance of the
818 `CIM_ProtocolService`. The mapping is implemented as an invocation of the `RequestStateChange()`
819 method on the instance.

820 6.7.2.1.1 Command Form

```
821 reset <CIM_ProtocolService single instance>
```

822 6.7.2.1.2 CIM Requirements

```
823 uint16 EnabledState;
824 uint16 RequestedState;
825 uint32 EnabledLogicalElement.RequestStateChange (
826     [IN] uint16 RequestedState,
827     [OUT] REF CIM_ConcreteJob Job,
828     [IN] datetime TimeoutPeriod );
```

829 6.7.2.1.3 Behavior Requirements

```
830 $instance=<CIM_ProtocolService single instance>
831 &smResetRSC ( $instance.getObjectPath() );
832 &smEnd;
```

833 6.7.3 Set

834 This section describes how to implement the `set` verb when it is applied to an instance of
835 `CIM_ProtocolService`. Implementations may support the use of the `set` verb with `CIM_ProtocolService`.

836 The `set` verb is used to modify descriptive properties of the `CIM_ProtocolService` instance.

837 6.7.3.1 General Usage of Set for a Single Property

838 This command form corresponds to the general usage of the `set` verb to modify a single property of a
839 target instance. This is the most common case.

840 The requirement for supporting modification of a property using this command form shall be equivalent to
841 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
842 in the [SSH Service Profile](#).

843 6.7.3.1.1 Command Form

```
844 set <CIM_ProtocolService single instance> <propertyname>=<propertyvalue>
```

845 6.7.3.1.2 CIM Requirements

846 See `CIM_ProtocolService` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
847 modifiable properties.

848 6.7.3.1.3 Behavior Requirements

```
849 $instance=<CIM_ProtocolService single instance>
850 #propertyName[] = {<propertyname>};
851 #propertyValues[] = {<propertyvalue>};
852 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
853 &smEnd;
```

854 6.7.3.2 General Usage of Set for Multiple Properties

855 This command form corresponds to the general usage of the `set` verb to modify multiple properties of a
856 target instance where there is not an explicit relationship between the properties. This is the most
857 common case.

858 The requirement for supporting modification of a property using this command form shall be equivalent to
859 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
860 in the [SSH Service Profile](#).

861 6.7.3.2.1 Command Form

```
862 set <CIM_ProtocolService single instance> <propertyname1>=<propertyvalue1>
863 <propertynamen>=<propertyvaluen>
```

864 6.7.3.2.2 CIM Requirements

865 See `CIM_ProtocolService` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
866 mandatory properties.

867 6.7.3.2.3 Behavior Requirements

```
868 $instance=<CIM_ProtocolService single instance>
869 #propertyName[] = {<propertyname>};
870 for #i < n
871 {
872     #propertyName[#i] = <propertyname#i>
873     #propertyValues[#i] = <propertyvalue#i>
874 }
875 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
876 &smEnd;
```

877 **6.7.4 Show**

878 This section describes how to implement the `show` verb when applied to an instance of
879 `CIM_ProtocolService`. Implementations shall support the use of the `show` verb with `CIM_ProtocolService`.

880 The `show` verb is used to display information about the `CIM_ProtocolService`.

881 **6.7.4.1 Show a Single Instance**

882 This command form is for the `show` verb applied to a single instance of `CIM_ProtocolService`.

883 **6.7.4.1.1 Command Form**

```
884 show <CIM_ProtocolService single instance>
```

885 **6.7.4.1.2 CIM Requirements**

886 See `CIM_ProtocolService` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
887 mandatory properties.

888 **6.7.4.1.3 Behavior Requirements**

889 **6.7.4.1.3.1 Preconditions**

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

891 **6.7.4.1.3.2 Pseudo Code**

```
892 $instance=<CIM_ProtocolService single instance>
893 #propertylist[] = NULL;
894 if (false == #all)
895     {
896         propertylist[] = { //all mandatory non-key properties };
897     }
898 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
899 &smEnd;
```

900 **6.7.4.2 Show Multiple Instances**

901 This command form is for the `show` verb applied to multiple instances of `CIM_ProtocolService`. This
902 command form corresponds to UFT-based selection within a scoping system.

903 **6.7.4.2.1 Command Form**

```
904 show <CIM_ProtocolService multiple instances>
```

905 **6.7.4.2.2 CIM Requirements**

906 See `CIM_ProtocolService` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
907 mandatory properties.

908 **6.7.4.2.3 Behavior Requirements**

909 **6.7.4.2.3.1 Preconditions**

910 `$containerInstance` contains the instance of `CIM_ComputerSystem` for which we are displaying
911 scoped instances of the `CIM_ProtocolService`. The [SSH Service Profile](#) requires that the

912 CIM_ProtocolService instance be associated with its scoping system via an instance of the
913 CIM_HostedService association.

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

915 **6.7.4.2.3.2 Pseudo Code**

```
916 #propertylist[] = NULL;
917 if (false == #all)
918 {
919     #propertylist[] = { //all mandatory non-key properties };
920 }
921 &smShowInstances ( "CIM_ProtocolService", "CIM_HostedService",
922     $containerInstance.getObjectPath(), #propertylist[] );
923 &smEnd;
```

924 **6.7.5 Start**

925 This section describes how to implement the `start` verb when applied to an instance of
926 CIM_ProtocolService. Implementations may support the use of the `start` verb with
927 CIM_ProtocolService.

928 The `start` verb is used to enable the CIM_ProtocolService.

929 **6.7.5.1 Start a Single Instance**

930 This command form is for the `start` verb applied to a single instance of CIM_ProtocolService.

931 **6.7.5.1.1 Command Form**

```
932 start <CIM_ProtocolService single instance>
```

933 **6.7.5.1.2 CIM Requirements**

```
934 uint16 EnabledState;
935 uint16 RequestedState;
936 uint32 EnabledLogicalElement.RequestStateChange (
937     [IN] uint16 RequestedState,
938     [OUT] REF CIM_ConcreteJob Job,
939     [IN] datetime TimeoutPeriod );
```

940 **6.7.5.1.3 Behavior Requirements**

```
941 $instance=<CIM_ProtocolService single instance>
942 &smStartRSC ( $instance.getObjectPath() );
943 &smEnd;
```

944 **6.7.6 Stop**

945 This section describes how to implement the `stop` verb when applied to an instance of
946 CIM_ProtocolService. Implementations may support the use of the `stop` verb with CIM_ProtocolService.

947 The `stop` verb is used to disable the CIM_ProtocolService.

948 **6.7.6.1 Stop a Single Instance**

949 This command form is for the `stop` verb applied to a single instance of CIM_ProtocolService.

950 **6.7.6.1.1 Command Form**

951 `stop <CIM_ProtocolService single instance>`

952 **6.7.6.1.2 CIM Requirements**

```
953 uint16 EnabledState;
954 uint16 RequestedState;
955 uint32 EnabledLogicalElement.RequestStateChange (
956     [IN] uint16 RequestedState,
957     [OUT] REF CIM_ConcreteJob Job,
958     [IN] datetime TimeoutPeriod );
```

959 **6.7.6.1.3 Behavior Requirements**

```
960 $instance=<CIM_ProtocolService single instance>
961 &smStopRSC ( $instance.getObjectPath() );
962 &smEnd;
```

963 **6.8 CIM_ServiceAccessBySAP**

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

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

970 **Table 9 – Command Verb Requirements for CIM_ServiceAccessBySAP**

| 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 | |

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

973 **6.8.1 Ordering of Results**

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

- 976 • Results for `CIM_ServiceAccessBySAP` are unordered; therefore, no algorithm is defined.

977 6.8.2 Show

978 This section describes how to implement the `show` verb when applied to an instance of
979 `CIM_ServiceAccessBySAP`. Implementations shall support the use of the `show` verb with
980 `CIM_ServiceAccessBySAP`.

981 The `show` command is used to display information about the `CIM_ServiceAccessBySAP` instance or
982 instances.

983 6.8.2.1 Show Multiple Instances – `CIM_ProtocolService` Reference

984 This command form is for the `show` verb applied to multiple instances. This command form corresponds
985 to a `show` command issued against `CIM_ServiceAccessBySAP` where only one reference is specified
986 and the reference is to an instance of `CIM_ProtocolService`.

987 6.8.2.1.1 Command Form

```
988 show <CIM_ServiceAccessBySAP multiple instances>
```

989 6.8.2.1.2 CIM Requirements

990 See `CIM_ServiceAccessBySAP` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
991 mandatory properties.

992 6.8.2.1.3 Behavior Requirements

993 6.8.2.1.3.1 Preconditions

994 `$instance` contains the instance of `CIM_ProtocolService` which is referenced by
995 `CIM_ServiceAccessBySAP`.

996 6.8.2.1.3.2 Pseudo Code

```
997 &smShowAssociationInstances ( "CIM_ServiceAccessBySAP", $instance.getObjectPath() );  
998 &smEnd;
```

999 6.8.2.2 Show Multiple Instances – `CIM_TCPProtocolEndpoint` Reference

1000 This command form is for the `show` verb applied to multiple instances. This command form corresponds
1001 to a `show` command issued against `CIM_ServiceAccessBySAP` where the reference specified is to an
1002 instance of `CIM_TCPProtocolEndpoint`.

1003 6.8.2.2.1 Command Form

```
1004 show <CIM_ServiceAccessBySAP multiple instances>
```

1005 6.8.2.2.2 CIM Requirements

1006 See `CIM_ServiceAccessBySAP` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
1007 mandatory properties.

1008 6.8.2.2.3 Behavior Requirements

1009 6.8.2.2.3.1 Preconditions

1010 `$instance` contains the instance of `CIM_TCPProtocolEndpoint` which is referenced by
1011 `CIM_ServiceAccessBySAP`.

1012 **6.8.2.2.3.2 Pseudo Code**

```
1013 &smShowAssociationInstances ( "CIM_ServiceAccessBySAP", $instance.getObjectPath() );
1014 &smEnd;
```

1015 **6.8.2.3 Show a Single Instance – Both References**

1016 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 1017 a `show` command issued against `CIM_ServiceAccessBySAP` where both references are specified and
 1018 therefore the desired instance is unambiguously identified.

1019 **6.8.2.3.1 Command Form**

```
1020 show <CIM_ServiceAccessBySAP single instance>
```

1021 **6.8.2.3.2 CIM Requirements**

1022 See `CIM_ServiceAccessBySAP` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
 1023 mandatory properties.

1024 **6.8.2.3.3 Behavior Requirements**

1025 **6.8.2.3.3.1 Preconditions**

1026 `$instanceA` contains the instance of `CIM_TCPProtocolEndpoint` which is referenced by
 1027 `CIM_ServiceAccessBySAP`.

1028 `$instanceB` contains the instance of `CIM_ProtocolService` which is referenced by
 1029 `CIM_ServiceAccessBySAP`.

1030 **6.8.2.3.3.2 Pseudo Code**

```
1031 &smShowAssociationInstance ( "CIM_ServiceAccessBySAP", $instanceA.getObjectPath(),
1032     $instanceB.getObjectPath() );
1033 &smEnd;
```

1034 **6.9 CIM_SSHCapabilities**

1035 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

1036 Table 10 lists each SM CLP verb, the required level of support for the verb in conjunction with instances
 1037 of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the
 1038 verb and target. Table 10 is for informational purposes only; in case of a conflict between Table 10 and
 1039 requirements detailed in the following sections, the text detailed in the following sections supersedes the
 1040 information in Table 10.

1041 **Table 10 – Command Verb Requirements for CIM_SSHCapabilities**

| 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 6.9.2. |
| start | Not supported | |
| stop | Not supported | |

1042 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,
1043 `reset`, `set`, `start`, and `stop`.

1044 6.9.1 Ordering of Results

1045 When results are returned for multiple instances of `CIM_SSHCapabilities`, implementations shall utilize
1046 the following algorithm to produce the natural (that is, default) ordering:

- 1047 • Results for `CIM_SSHCapabilities` are unordered; therefore, no algorithm is defined.

1048 6.9.2 Show

1049 This section describes how to implement the `show` verb when applied to an instance of
1050 `CIM_SSHCapabilities`. Implementations shall support the use of the `show` verb with
1051 `CIM_SSHCapabilities`.

1052 The `show` verb is used to display information about an instance or instances of the `CIM_SSHCapabilities`
1053 class.

1054 6.9.2.1 Show a Single Instance

1055 This command form is for the `show` verb applied to a single instance of `CIM_SSHCapabilities`.

1056 6.9.2.1.1 Command Form

```
1057 show <CIM_SSHCapabilities single instance>
```

1058 6.9.2.1.2 CIM Requirements

1059 See `CIM_SSHCapabilities` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
1060 mandatory properties.

1061 6.9.2.1.3 Behavior Requirements

1062 6.9.2.1.3.1 Preconditions

1063 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1064 6.9.2.1.3.2 Pseudo Code

```
1065 $instance=<CIM_SSHCapabilities single instance>
1066 #propertylist[] = NULL;
1067 if ( false == #all)
1068 {
1069     #propertylist[] = { //all mandatory non-key properties }
1070 }
1071 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
1072 &smEnd;
```

1073 **6.9.2.2 Show Multiple Instances**

1074 This command form is for the `show` verb applied to multiple instances of `CIM_SSHCapabilities`. This
 1075 command form corresponds to UFsT-based selection within a capabilities collection.

1076 **6.9.2.2.1 Command Form**

1077 `show <CIM_SSHCapabilities multiple instances>`

1078 **6.9.2.2.2 CIM Requirements**

1079 See `CIM_SSHCapabilities` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
 1080 mandatory properties.

1081 **6.9.2.2.3 Behavior Requirements**

1082 **6.9.2.2.3.1 Preconditions**

1083 `$containerInstance` contains the instance of `CIM_ConcreteCollection` for which contained
 1084 `CIM_Capabilities` instances are displayed. `CIM_Capabilities` instances are addressed via an aggregating
 1085 instance of `CIM_ConcreteCollection`.

1086 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1087 **6.9.2.2.3.2 Pseudo Code**

```
1088 #propertylist[] = NULL;
1089 if ( false == #all)
1090     {
1091         #propertylist[] = { //all mandatory non-key properties }
1092     }
1093 &smShowInstances ( "CIM_SSHCapabilities", "CIM_MemberOfCollection",
1094     $containerInstance.GetObjectPath(), #propertylist[] );
1095 &smEnd;
```

1096 **6.10 CIM_SSHProtocolEndpoint**

1097 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

1098 Table 11 lists each SM CLP verb, the required level of support for the verb in conjunction with instances
 1099 of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the
 1100 verb and target. Table 11 is for informational purposes only; in case of a conflict between Table 11 and
 1101 requirements detailed in the following sections, the text detailed in the following sections supersedes the
 1102 information in Table 11.

1103 **Table 11 – Command Verb Requirements for CIM_SSHProtocolEndpoint**

| Command Verb | Requirement | Comments |
|--------------|---------------|-------------|
| create | Not supported | |
| delete | Not supported | |
| dump | Not supported | |
| load | Not supported | |
| reset | Not supported | |
| set | May | See 6.10.2. |

| Command Verb | Requirement | Comments |
|--------------|---------------|-------------|
| show | Shall | See 6.10.3. |
| start | Not supported | |
| stop | May | See 6.10.4. |

1104 No mapping is defined for the following verbs for the specified target: `dump` and `load`.

1105 6.10.1 Ordering of Results

1106 When results are returned for multiple instances of `CIM_SSHProtocolEndpoint`, implementations shall
1107 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1108 • Results for `CIM_SSHProtocolEndpoint` are unordered; therefore, no algorithm is defined.

1109 6.10.2 Set

1110 This section describes how to implement the `set` verb when it is applied to an instance of
1111 `CIM_SSHProtocolEndpoint`. Implementations may support the use of the `set` verb with
1112 `CIM_SSHProtocolEndpoint`.

1113 The `set` verb is used to modify descriptive properties of the `CIM_SSHProtocolEndpoint` instance.

1114 6.10.2.1 General Usage of Set for a Single Property

1115 This command form corresponds to the general usage of the `set` verb to modify a single property of a
1116 target instance. This is the most common case.

1117 The requirement for supporting modification of a property using this command form shall be equivalent to
1118 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
1119 in the [SSH Service Profile](#).

1120 6.10.2.1.1 Command Form

```
1121 set <CIM_SSHProtocolEndpoint single instance> <propertyname>=<propertyvalue>
```

1122 6.10.2.1.2 CIM Requirements

1123 See `CIM_SSHProtocolEndpoint` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
1124 modifiable properties.

1125 6.10.2.1.3 Behavior Requirements

```
1126 $instance=<CIM_SSHProtocolEndpoint single instance>
1127 #propertyNames[] = {<propertyname>};
1128 #propertyValues[] = {<propertyvalue>};
1129 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
1130 &smEnd;
```

1131 6.10.2.2 General Usage of Set for Multiple Properties

1132 This command form corresponds to the general usage of the `set` verb to modify multiple properties of a
1133 target instance where there is not an explicit relationship between the properties. This is the most
1134 common case.

1135 The requirement for supporting modification of a property using this command form shall be equivalent to
 1136 the requirement for supporting modification of the property using the ModifyInstance operation as defined
 1137 in the [SSH Service Profile](#).

1138 6.10.2.2.1 Command Form

```
1139 set <CIM_SSHProtocolEndpoint single instance> <propertyname1>=<propertyvalue1>
1140 <propertynamen>=<propertyvaluen>
```

1141 6.10.2.2.2 CIM Requirements

1142 See CIM_SSHProtocolEndpoint in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
 1143 mandatory properties.

1144 6.10.2.2.3 Behavior Requirements

```
1145 $instance=<CIM_SSHProtocolEndpoint single instance>
1146 #propertyNames[] = {<propertyname>};
1147 for #i < n
1148 {
1149     #propertyNames[#i] = <propertyname#i>
1150     #propertyValues[#i] = <propertyvalue#i>
1151 }
1152 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
1153 &smEnd;
```

1154 6.10.3 Show

1155 This section describes how to implement the `show` verb when applied to an instance of
 1156 CIM_SSHProtocolEndpoint. Implementations shall support the use of the `show` verb with
 1157 CIM_SSHProtocolEndpoint.

1158 The `show` verb is used to display information about an SSH session.

1159 Note that CIM_BindsTo and CIM_HostedAccessPoint are both Addressing Associations. Thus, an
 1160 implementation of the SM CLP has a choice when exposing the address for an instance of
 1161 CIM_SSHProtocolEndpoint. For completeness, mappings are shown for both associations, though only
 1162 one would be applicable in a given implementation.

1163 6.10.3.1 Show a Single Instance

1164 This command form is for the `show` verb applied to a single instance of CIM_SSHProtocolEndpoint.

1165 6.10.3.1.1 Command Form

```
1166 show <CIM_SSHProtocolEndpoint single instance>
```

1167 6.10.3.1.2 CIM Requirements

1168 See CIM_SSHProtocolEndpoint in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
 1169 mandatory properties.

1170 6.10.3.1.3 Behavior Requirements

1171 6.10.3.1.3.1 Preconditions

1172 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1173 **6.10.3.1.3.2 Pseudo Code**

```

1174 $instance=<CIM_SSHProtocolEndpoint single instance>
1175 #propertylist[] = NULL;
1176 if ( false == #all)
1177     {
1178         #propertylist[] = { //all mandatory non-key properties }
1179     }
1180 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
1181 &smEnd;

```

1182 **6.10.3.2 Show Multiple Instances Scoped by System**

1183 This command form is for the `show` verb applied to multiple instances of `CIM_SSHProtocolEndpoint`. This
 1184 command form corresponds to UFsT-based selection within a scoping system.

1185 **6.10.3.2.1 Command Form**

```

1186 show <CIM_SSHProtocolEndpoint multiple instances>

```

1187 **6.10.3.2.2 CIM Requirements**

1188 See `CIM_SSHProtocolEndpoint` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
 1189 mandatory properties.

1190 **6.10.3.2.3 Behavior Requirements**1191 **6.10.3.2.3.1 Preconditions**

1192 `$containerInstance` contains the instance of `CIM_ComputerSystem` for which scoped endpoints
 1193 (`CIM_SSHProtocolEndpoint` instances) are displayed. The [SSH Service Profile](#) requires that the
 1194 `CIM_SSHProtocolEndpoint` instance be associated with its scoping system via an instance of the
 1195 `CIM_HostedAccessPoint` association.

1196 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1197 **6.10.3.2.3.2 Pseudo Code**

```

1198 #propertylist[] = NULL;
1199 if (false == #all)
1200     {
1201         #propertylist[] = { //all mandatory non-key properties }
1202     }
1203 &smShowInstances ( "CIM_SSHProtocolEndpoint", "CIM_HostedAccessPoint",
1204     $containerInstance.getObjectPath(), #propertylist[] );
1205 &smEnd;

```

1206 **6.10.3.3 Show Multiple Instances Scoped by a TCPProtocolEndpoint**

1207 This command form is for the `show` verb applied to multiple instances of `CIM_SSHProtocolEndpoint`. This
 1208 command form corresponds to UFsT-based selection within a scoping `CIM_TCPProtocolEndpoint`
 1209 instance.

1210 **6.10.3.3.1 Command Form**

```

1211 show <CIM_SSHProtocolEndpoint multiple instances>

```


1212 6.10.3.3.2 CIM Requirements

1213 See CIM_SSHProtocolEndpoint in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
1214 mandatory properties.

1215 6.10.3.3.3 Behavior Requirements

1216 6.10.3.3.3.1 Preconditions

1217 \$containerInstance contains the instance of CIM_TCPProtocolEndpoint for which scoped endpoints
1218 (CIM_SSHProtocolEndpoint instances) are displayed. The [SSH Service Profile](#) requires that the
1219 CIM_SSHProtocolEndpoint instance be associated with a CIM_TCPProtocolEndpoint instance via an
1220 instance of the CIM_BindsTo association.

1221 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1222 6.10.3.3.3.2 Pseudo Code

```
1223 #propertylist[] = NULL;
1224 if (false == #all)
1225     {
1226         #propertylist[] = { //all mandatory non-key properties };
1227     }
1228 &smShowInstances ( "CIM_SSHProtocolEndpoint", "CIM_BindsTo",
1229     $containerInstance.getObjectPath(), #propertylist[] );
1230 &smEnd;
```

1231 6.10.4 Stop

1232 This section describes how to implement the `stop` verb when applied to an instance of
1233 CIM_SSHProtocolEndpoint. Implementations may support the use of the `stop` verb with
1234 CIM_SSHProtocolEndpoint.

1235 The `stop` verb is used to disable an endpoint.

1236 6.10.4.1 Stop a Single Instance

1237 This command form is for the `stop` verb applied to a single instance of CIM_SSHProtocolEndpoint. The
1238 lifecycle of an SSH session corresponds to the lifecycle of the CIM_SSHProtocolEndpoint which
1239 represents it. Therefore, stopping an SSH service corresponds to a delete of the underlying instance.

1240 6.10.4.1.1 Command Form

```
1241 stop <CIM_SSHProtocolEndpoint single instance>
```

1242 6.10.4.1.2 CIM Requirements

1243 See CIM_SSHProtocolEndpoint in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
1244 mandatory properties.

1245 6.10.4.1.3 Behavior Requirements

```
1246 $instance=<CIM_SSHProtocolEndpoint single instance>
1247 &smDeleteInstance ( $instance.getObjectPath() );
1248 &smEnd;
```

1249 **6.11 CIM_SSHSettingData**1250 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

1251 Table 12 lists each SM CLP verb, the required level of support for the verb in conjunction with instances
 1252 of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the
 1253 verb and target. Table 12 is for informational purposes only; in case of a conflict between Table 12 and
 1254 requirements detailed in the following sections, the text detailed in the following sections supersedes the
 1255 information in Table 12.

1256 **Table 12 – Command Verb Requirements for CIM_SSHSettingData**

| Command Verb | Requirement | Comments |
|--------------|---------------|-------------|
| create | Not supported | |
| delete | Not supported | |
| dump | Not supported | |
| load | Not supported | |
| reset | Not supported | |
| set | May | See 6.11.2. |
| show | Shall | See 6.11.3. |
| start | Not supported | |
| stop | Not supported | |

1257 No mapping is defined for the following verbs for the specified target: `dump` and `load`.1258 **6.11.1 Ordering of Results**

1259 When results are returned for multiple instances of `CIM_SSHSettingData`, implementations shall utilize
 1260 the following algorithm to produce the natural (that is, default) ordering:

- 1261 • Results for `CIM_SSHSettingData` are unordered; therefore, no algorithm is defined.

1262 **6.11.2 Set**

1263 This section describes how to implement the `set` verb when it is applied to an instance of
 1264 `CIM_SSHSettingData`. Implementations may support the use of the `set` verb with `CIM_SSHSettingData`.

1265 The `set` verb is used to modify a configuration represented by an instance of `CIM_SSHSettingData`.1266 **6.11.2.1 General Usage of Set for a Single Property**

1267 This command form corresponds to the general usage of the `set` verb to modify a single property of a
 1268 target instance. This is the most common case.

1269 The requirement for supporting modification of a property using this command form shall be equivalent to
 1270 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
 1271 in the [SSH Service Profile](#).

1272 **6.11.2.1.1 Command Form**1273 `set <CIM_SSHSettingData single instance> <propertyname>=<propertyvalue>`

1274 6.11.2.1.2 CIM Requirements

1275 See CIM_SSHSettingData in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
1276 modifiable properties.

1277 6.11.2.1.3 Behavior Requirements

```
1278 $instance=<CIM_SSHSettingData single instance>
1279 #propertyName[] = {<propertyname>};
1280 #propertyValues[] = {<propertyvalue>};
1281 &smSetInstancez ( $instance, #propertyName[], #propertyValues[] );
1282 &smEnd;
```

1283 6.11.2.2 General Usage of Set for Multiple Properties

1284 This command form corresponds to the general usage of the `set` verb to modify multiple properties of a
1285 target instance where there is not an explicit relationship between the properties. This is the most
1286 common case.

1287 The requirement for supporting modification of a property using this command form shall be equivalent to
1288 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
1289 in the [SSH Service Profile](#).

1290 6.11.2.2.1 Command Form

```
1291 set <CIM_SSHSettingData single instance> <propertyname1>=<propertyvalue1>
1292 <propertynamen>=<propertyvaluen>
```

1293 6.11.2.2.2 CIM Requirements

1294 See CIM_SSHSettingData in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
1295 mandatory properties.

1296 6.11.2.2.3 Behavior Requirements

```
1297 $instance=<CIM_SSHSettingData single instance>
1298 #propertyName[] = {<propertyname>};
1299 for #i < n
1300 {
1301     #propertyName[#i] = <propertyname#i>
1302     #propertyValues[#i] = <propertyvalue#i>
1303 }
1304 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
1305 &smEnd;
```

1306 6.11.3 Show

1307 This section describes how to implement the `show` verb when applied to an instance of
1308 CIM_SSHSettingData. Implementations shall support the use of the `show` verb with
1309 CIM_SSHSettingData.

1310 The `show` verb is used to display information about the CIM_SSHSettingData instance.

1311 6.11.3.1 Show a Single Instance

1312 This command form is for the `show` verb applied to a single instance of CIM_SSHSettingData.

1313 **6.11.3.1.1 Command Form**1314

```
show <CIM_SSHSettingData single instance>
```

1315 **6.11.3.1.2 CIM Requirements**1316 See CIM_SSHSettingData in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
1317 mandatory properties.1318 **6.11.3.1.3 Behavior Requirements**1319 **6.11.3.1.3.1 Preconditions**

1320 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1321 **6.11.3.1.3.2 Pseudo Code**1322

```
$instance=<CIM_SSHSettingData single instance>  
1323 &lShowTCPEndpoint ( $instance, #all );  
1324 &smEnd;
```

1325 **6.11.3.2 Show Multiple Instances Scoped by ConcreteCollection**1326 This command form is for the show verb applied to multiple instances of CIM_SSHSettingData. This
1327 command form corresponds to UFsT-based selection within an instance of CIM_ConcreteCollection.1328 **6.11.3.2.1 Command Form**1329

```
show <CIM_SSHSettingData multiple instances>
```

1330 **6.11.3.2.2 CIM Requirements**1331 See CIM_SSHSettingData in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
1332 mandatory properties.1333 **6.11.3.2.3 Behavior Requirements**1334 **6.11.3.2.3.1 Preconditions**1335 \$containerInstance contains the instance of CIM_ConcreteCollection for which contained
1336 CIM_SSHSettingData instances are displayed. The [SMASH Collections Profile](#) requires that the
1337 CIM_SSHSettingData instances be aggregated into an addressing collection via
1338 CIM_MemberOfCollection.

1339 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1340 **6.11.3.2.3.2 Pseudo Code**1341

```
#propertylist[] = NULL;  
1342 //this property list will match the property list in lShowTCPEndpoint()  
1343 if (false == #all)  
1344 {  
1345     #propertylist[] = { //all mandatory non-key properties }  
1346 }  
1347 &smShowInstances ( "CIM_SSHSettingData", "CIM_MemberOfCollection",  
1348     $containerInstance.getObjectPath(), #propertylist[] );  
1349 &smEnd;
```

1350 6.12 CIM_TCPProtocolEndpoint

1351 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

1352 Table 13 lists each SM CLP verb, the required level of support for the verb in conjunction with instances
 1353 of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the
 1354 verb and target. Table 13 is for informational purposes only; in case of a conflict between Table 13 and
 1355 requirements detailed in the following sections, the text detailed in the following sections supersedes the
 1356 information in Table 13.

1357 **Table 13 – Command Verb Requirements for CIM_TCPProtocolEndpoint**

| Command Verb | Requirement | Comments |
|--------------|---------------|-------------|
| create | May | See 6.12.2. |
| delete | May | See 6.12.3. |
| dump | Not supported | |
| load | Not supported | |
| reset | Not supported | |
| set | May | See 6.12.4. |
| show | Shall | See 6.12.5. |
| start | Not supported | |
| stop | Not supported | |

1358 No mapping is defined for the following verbs for the specified target: `dump` and `load`.

1359 6.12.1 Ordering of Results

1360 When results are returned for multiple instances of CIM_TCPProtocolEndpoint, implementations shall
 1361 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1362 • Results for CIM_TCPProtocolEndpoint are unordered; therefore, no algorithm is defined.

1363 6.12.2 Create

1364 This section describes how to implement the `create` verb when applied to an instance of
 1365 CIM_TCPProtocolEndpoint. Implementations may support the use of the `create` verb with
 1366 CIM_TCPProtocolEndpoint.

1367 The `create` verb is used to create an additional CIM_TCPProtocolEndpoint instance representing a port
 1368 upon which the SSH service is listening.

1369 6.12.2.1 Create Specifying the Required Port Number

1370 In order to create an instance of CIM_TCPProtocolEndpoint, a client is required to supply the desired IP
 1371 port.

1372 6.12.2.1.1 Command Form

1373 `create <CIM_TCPProtocolEndpoint single instance> portnumber=<desiredport>`

1374 **6.12.2.1.2 CIM Requirements**

1375 See CIM_TCPProtocolEndpoint in the “CIM Elements” section of the [SSH Service Profile](#) for the
1376 CIM_ProtocolService.AddListeningPort property.

1377 **6.12.2.1.3 Behavior Requirements**1378 **6.12.2.1.3.1 Preconditions**

1379 \$Service contains the CIM_ProtocolService instance for which a new endpoint is created.

1380 **6.12.2.1.3.2 Pseudo Code**

```
1381 // container instance specified in the Resultant Address
1382 //the desired address is required, if it is not specified, fail
1383 if (NULL == <desiredport>) {
1384     $OperationError = smNewInstance("CIM_Error");
1385     //CIM_ERR_FAILED
1386     $OperationError.CIMStatusCode = 1;
1387     //Software Error
1388     $OperationError.ErrorType = 4;
1389     //Unknown
1390     $OperationError.PerceivedSeverity = 0;
1391     $OperationError.OwningEntity = DMTF:SMCLP;
1392     $OperationError.MessageID = 0x0000000D;
1393     $OperationError.Message = "A required property was not specified.";
1394     &smAddError($job, $OperationError);
1395     &smMakeCommandStatus($job);
1396     &smEnd;
1397 }
1398 $Endpoint = smNewInstance ("CIM_TCPProtocolEndpoint");
1399 //build the parameter lists and invoke the method
1400 %InArguments[] = {newArgument("PortNumber", <desiredport>}
1401 %OutArguments[] = { newArgument("Endpoint",
1402     $Endpoint.GetObjectPath() ) };
1403 //invoke method
1404 #returnStatus = smOpInvokeMethod ($Service.GetObjectPath(),
1405     "AddListeningEndpoint",
1406     %InArguments[],
1407     %OutArguments[]);
1408 // process return code to CLP Command Status
1409 if (0 != #Error.code) {
1410     //method invocation failed
1411     if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) ) {
1412         // if the method invocation contains an embedded error
1413         // use it for the Error for the overall job
1414         &smAddError($job, #Error.$error[0]);
1415         &smMakeCommandStatus($job);
1416         &smEnd;
1417     }
1418     else if (#Error.code == 17) {
1419         //trap for CIM_METHOD_NOT_FOUND
```

```
1420 //and make nice Unsupported msg.
1421 //unsupported
1422 $OperationError = smNewInstance("CIM_Error");
1423 //CIM_ERR_NOT_SUPPORTED
1424 $OperationError.CIMStatusCode = 7;
1425 //Other
1426 $OperationError.ErrorType = 1;
1427 //Low
1428 $OperationError.PerceivedSeverity = 2;
1429 $OperationError.OwningEntity = DMTF:SMCLP;
1430 $OperationError.MessageID = 0x00000001;
1431 $OperationError.Message = "Operation is not supported.";
1432 &smAddError($job, $OperationError);
1433 &smMakeCommandStatus($job);
1434 &smEnd;
1435 }
1436 else {
1437 //operation failed, but no detailed error instance, need to make one up
1438 //make an Error instance and associate with job for Operation
1439 $OperationError = smNewInstance("CIM_Error");
1440 //CIM_ERR_FAILED
1441 $OperationError.CIMStatusCode = 1;
1442 //Software Error
1443 $OperationError.ErrorType = 4;
1444 //Unknown
1445 $OperationError.PerceivedSeverity = 0;
1446 $OperationError.OwningEntity = DMTF:SMCLP;
1447 $OperationError.MessageID = 0x00000009;
1448 $OperationError.Message = "An internal software error has occurred.";
1449 &smAddError($job, $OperationError);
1450 &smMakeCommandStatus($job);
1451 &smEnd;
1452 }
1453 }//if CIM op failed
1454 else if (0 == #returnStatus) {
1455 //completed successfully
1456 &lShowTCPEndpoint($Endpoint, "false");
1457 &smEnd;
1458 }
1459 else if (4 == #returnStatus) {
1460 //generic failure
1461 $OperationError = smNewInstance("CIM_Error");
1462 //CIM_ERR_FAILED
1463 $OperationError.CIMStatusCode = 1;
1464 //Other
1465 $OperationError.ErrorType = 1;
1466 //Low
1467 $OperationError.PerceivedSeverity = 2;
1468 $OperationError.OwningEntity = DMTF:SMCLP;
```

```

1469     $OperationError.MessageID = 0x00000002;
1470     $OperationError.Message = "Failed. No further information is available.";
1471     &smAddError($job, $OperationError);
1472     &smMakeCommandStatus($job);
1473 }
1474 else {
1475     //invalid parameter
1476     $OperationError = smNewInstance("CIM_Error");
1477     //CIM_ERR_FAILED
1478     $OperationError.CIMStatusCode = 1;
1479     //Other
1480     $OperationError.ErrorType = 1;
1481     //Low
1482     $OperationError.PerceivedSeverity = 2;
1483     $OperationError.OwningEntity = DMTF:SMCLP;
1484     $OperationError.MessageID = 0x00000004;
1485     $OperationError.Message = "One or more parameters specified are invalid.";
1486     &smAddError($job, $OperationError);
1487     &smMakeCommandStatus($job);
1488     &smEnd;
1489 }

```

1490 6.12.3 Delete

1491 This section describes how to implement the `delete` verb when applied to an instance of
 1492 `CIM_TCPProtocolEndpoint`. Implementations may support the use of the `delete` verb with
 1493 `CIM_TCPProtocolEndpoint`.

1494 The `delete` command is used to remove an instance of `CIM_TCPProtocolEndpoint` which represents a
 1495 virtual MAC.

1496 6.12.3.1 Delete a Single Instance

1497 Delete a single instance of `CIM_TCPProtocolEndpoint`.

1498 6.12.3.1.1 Command Form

```
1499 delete <CIM_TCPProtocolEndpoint single instance>
```

1500 6.12.3.1.2 CIM Requirements

1501 See `CIM_TCPProtocolEndpoint` in the "CIM Elements" section of the [SSH Service Profile](#) for the
 1502 `CIM_TCPProtocolEndpoint` property.

1503 6.12.3.1.3 Behavior Requirements

```

1504 $instance=<CIM_TCPProtocolEndpoint single instance>
1505 &smOpDeleteInstance ( $instance.GetObjectPath() );
1506 &smEnd;

```


1507 6.12.4 Set

1508 This section describes how to implement the `set` verb when it is applied to an instance of
1509 `CIM_TCPProtocolEndpoint`. Implementations may support the use of the `set` verb with
1510 `CIM_TCPProtocolEndpoint`.

1511 The `set` verb is used to modify descriptive properties of the `CIM_TCPProtocolEndpoint` instance.

1512 6.12.4.1 General Usage of Set for a Single Property

1513 This command form corresponds to the general usage of the `set` verb to modify a single property of a
1514 target instance. This is the most common case.

1515 The requirement for supporting modification of a property using this command form shall be equivalent to
1516 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
1517 in the [SSH Service Profile](#).

1518 6.12.4.1.1 Command Form

```
1519 set <CIM_TCPProtocolEndpoint single instance> <propertyname>=<propertyvalue>
```

1520 6.12.4.1.2 CIM Requirements

1521 See `CIM_TCPProtocolEndpoint` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
1522 modifiable properties.

1523 6.12.4.1.3 Behavior Requirements

```
1524 $instance=<CIM_TCPProtocolEndpoint single instance>
1525 #propertyNames[] = {<propertyname>};
1526 #propertyValues[] = {<propertyvalue>};
1527 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
1528 &smEnd;
```

1529 6.12.4.2 General Usage of Set for Multiple Properties

1530 This command form corresponds to the general usage of the `set` verb to modify multiple properties of a
1531 target instance where there is not an explicit relationship between the properties. This is the most
1532 common case.

1533 The requirement for supporting modification of a property using this command form shall be equivalent to
1534 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
1535 in the [SSH Service Profile](#).

1536 6.12.4.2.1 Command Form

```
1537 set <CIM_TCPProtocolEndpoint single instance> <propertyname1>=<propertyvalue1>
1538 <propertynamen>=<propertyvaluen>
```

1539 6.12.4.2.2 CIM Requirements

1540 See `CIM_TCPProtocolEndpoint` in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
1541 mandatory properties.

1542 **6.12.4.2.3 Behavior Requirements**

```

1543 $instance=<CIM_TCPProtocolEndpoint single instance>
1544 #propertyNames[] = {<propertyname>};
1545 for #i < n
1546     {
1547         #propertyNames[#i] = <propertyname#i>
1548         #propertyValues[#i] = <propertyvalue#i>
1549     }
1550 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
1551 &smEnd;

```

1552 **6.12.5 Show**

1553 This section describes how to implement the `show` verb when applied to an instance of
 1554 CIM_TCPProtocolEndpoint. Implementations shall support the use of the `show` verb with
 1555 CIM_TCPProtocolEndpoint.

1556 The `show` verb is used to display information about a CIM_TCPProtocolEndpoint instance.

1557 **6.12.5.1 Show a Single Instance**

1558 This command form is for the `show` verb applied to a single instance of CIM_TCPProtocolEndpoint.

1559 **6.12.5.1.1 Command Form**

```

1560 show <CIM_TCPProtocolEndpoint single instance>

```

1561 **6.12.5.1.2 CIM Requirements**

1562 See CIM_TCPProtocolEndpoint in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
 1563 mandatory properties.

1564 **6.12.5.1.3 Behavior Requirements**1565 **6.12.5.1.3.1 Preconditions**

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

1567 **6.12.5.1.3.2 Pseudo Code**

```

1568 $instance=<CIM_TCPProtocolEndpoint single instance>
1569 &lShowTCPEndpoint ( $instance, #all );
1570 &smEnd;

```

1571 **6.12.5.2 Show Multiple Instances Scoped by a System**

1572 This command form is for the `show` verb applied to multiple instances of CIM_TCPProtocolEndpoint. This
 1573 command form corresponds to UFsT-based selection within a scoping system.

1574 **6.12.5.2.1 Command Form**

```

1575 show <CIM_TCPProtocolEndpoint multiple instances>

```

1576 6.12.5.2.2 CIM Requirements

1577 See CIM_TCPProtocolEndpoint in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
1578 mandatory properties.

1579 6.12.5.2.3 Behavior Requirements

1580 6.12.5.2.3.1 Preconditions

1581 \$containerInstance contains the instance of CIM_ComputerSystem for which we are displaying
1582 scoped endpoints (CIM_TCPProtocolEndpoint instances). The [SSH Service Profile](#) requires that the
1583 CIM_TCPProtocolEndpoint instance be associated with its scoping system via an instance of the
1584 CIM_HostedAccessPoint association.

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

1586 6.12.5.2.3.2 Pseudo Code

```
1587 #propertylist[] = NULL;
1588 //this property list will match the property list in lShowTCPEndpoint()
1589 if (false == #all)
1590 {
1591     #propertylist[] = { //all mandatory non-key properties };
1592 }
1593 &smShowInstances ( "CIM_TCPProtocolEndpoint", "CIM_HostedAccessPoint",
1594     $containerInstance.getObjectPath(), #propertylist[] );
1595 &smEnd;
```

1596 6.12.5.2.4 Show Multiple Instances Scoped by a ProtocolService

1597 This command form is for the show verb applied to multiple instances of CIM_TCPProtocolEndpoint. This
1598 command form corresponds to UFST-based selection within a scoping ProtocolService instance.

1599 6.12.5.2.5 Command Form

```
1600 show <CIM_TCPProtocolEndpoint multiple instances>
```

1601 6.12.5.2.6 CIM Requirements

1602 See CIM_TCPProtocolEndpoint in the “CIM Elements” section of the [SSH Service Profile](#) for the list of
1603 mandatory properties.

1604 6.12.5.2.7 Behavior Requirements

1605 6.12.5.2.7.1 Preconditions

1606 \$containerInstance contains the instance of CIM_ProtocolService for which we are displaying
1607 associated endpoints (CIM_TCPProtocolEndpoint instances). The [SSH Service Profile](#) requires that the
1608 CIM_TCPProtocolEndpoint instance be associated with an instance of CIM_ProtocolService via an
1609 instance of CIM_ServiceAccessBySAP.

1610 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1611 **6.12.5.2.7.2 Pseudo Code**

```
1612 #propertylist[] = NULL;
1613 //this property list will match the property list in lShowTCPEndpoint()
1614 if (false == #all)
1615     {
1616         #propertylist[] = { //all mandatory non-key properties };
1617     }
1618 &smShowInstances ( "CIM_TCPProtocolEndpoint", "CIM_ServiceAccessBySAP",
1619     $containerInstance.getObjectPath(), #propertylist[] );
1620 &smEnd;
```

1621 **6.12.5.3 Show Multiple Instances Scoped by a ProtocolEndpoint**

1622 This command form is for the show verb applied to multiple instances of CIM_TCPProtocolEndpoint. This
1623 command form corresponds to UFsT-based selection within a scoping CIM_ProtocolEndpoint instance
1624 with which the CIM_TCPProtocolEndpoint instances are associated via instances of CIM_BindsTo.

1625 **6.12.5.3.1 Command Form**

```
1626 show <CIM_TCPProtocolEndpoint multiple instances>
```

1627 **6.12.5.3.2 CIM Requirements**

1628 See CIM_TCPProtocolEndpoint in the "CIM Elements" section of the [SSH Service Profile](#) for the list of
1629 mandatory properties.

1630 **6.12.5.3.3 Behavior Requirements**1631 **6.12.5.3.3.1 Preconditions**

1632 \$containerInstance contains the instance of CIM_ProtocolEndpoint for which we are displaying
1633 associated endpoints (CIM_TCPProtocolEndpoint instances). The [SSH Service Profile](#) indicates that the
1634 CIM_TCPProtocolEndpoint instance can be associated with an instance of CIM_ProtocolEndpoint via an
1635 instance of CIM_BindsTo.

1636 #all is true if the "-all" option was specified with the command; otherwise, #all is false.

1637 **6.12.5.3.3.2 Pseudo Code**

```
1638 #propertylist[] = NULL;
1639 //this property list will match the property list in lShowTCPEndpoint()
1640 if (false == #all)
1641     {
1642         #propertylist[] = { //all mandatory non-key properties };
1643     }
1644 &smShowInstances ( "CIM_TCPProtocolEndpoint", "CIM_BindsTo",
1645     $containerInstance.getObjectPath(), #propertylist[] );
1646 &smEnd;
```

1647

1648
 1649
 1650
 1651
 1652

ANNEX A
 (informative)

Change Log

| Version | Date | Author | Description |
|---------|------------|--------|-----------------------|
| 1.0.0 | 2009-07-14 | | DMTF Standard Release |
| | | | |
| | | | |
| | | | |

1653