1

5 Telnet Service Profile SM CLP Command
6 Mapping Specification

10

33

CONTENTS

Tables

75

76   Foreword

77   The *Telnet Service Profile SM CLP Command Mapping Specification* (DSP0820) was prepared by the
78   Server Management Working Group.

Conventions

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

Acknowledgements

83   The authors wish to acknowledge the following participants from the DTMF 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   •   Aaron Merkin – IBM

91   •   Perry Vincent – Intel

92   •   John Leung – Intel

93

94   Introduction

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

99   The target audience for this specification is implementers of the SM CLP support for the *Telnet Service*
100   *Profile*.

101 # Telnet Service Profile SM CLP Command Mapping
102 # Specification

103 ## 1   Scope

104 This specification contains the requirements for an implementation of the SM CLP to provide access to,
105 and implement the behaviors of, the *Telnet Service Profile*.

106 ## 2   Normative References

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

110 ### 2.1   Approved References

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

113 DMTF DSP1006, *SMASH Collections Profile1.0,*
114 http://www.dmtf.org/standards/published_documents/DSP1006_1.0.pdf

115 DMTF DSP1016, *Telnet Service Profile 1.0*,
116 http://www.dmtf.org/standards/published_documents/DSP1016_1.0.pdf

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

119 ### 2.2   Other References

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

122 ## 3   Terms and Definitions

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

124 **3.1**
125 **can**
126 used for statements of possibility and capability, whether material, physical, or causal

127 **3.2**
128 **cannot**
129 used for statements of possibility and capability, whether material, physical or causal

130 **3.3**
131 **conditional**
132 indicates requirements to be followed strictly in order to conform to the document when the specified
133 conditions are met

134 **3.4**
135 **mandatory**
136 indicates requirements to be followed strictly in order to conform to the document and from which no
137 deviation is permitted

138 **3.5**
139 **may**
140 indicates a course of action permissible within the limits of the document

141 **3.6**
142 **need not**
143 indicates a course of action permissible within the limits of the document

144 **3.7**
145 **optional**
146 indicates a course of action permissible within the limits of the document

147 **3.8**
148 **shall**
149 indicates requirements to be followed strictly in order to conform to the document and from which no
150 deviation is permitted

151 **3.9**
152 **shall not**
153 indicates requirements to be followed strictly in order to conform to the document and from which no
154 deviation is permitted

155 **3.10**
156 **should**
157 indicates that among several possibilities, one is recommended as particularly suitable, without
158 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required

159 **3.11**
160 **should not**
161 indicates that a certain possibility or course of action is deprecated but not prohibited

162 # 4   Symbols and Abbreviated Terms

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

164 **4.1**
165 **CIM**
166 Common Information Model

167 **4.2**
168 **CLP**
169 Command Line Protocol

170 **4.3**
171 **DMTF**
172 Distributed Management Task Force

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

176 **4.5**
177 **SM**
178 Server Management

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

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

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

188 # 5   Recipes

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

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

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

204 **Table 1 – Local Recipes**

Recipe Name	Description	Definition
lShowTCPEndpoint	Show an instance of CIM_TCPProtocolEndpoint	See 5.1.

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

206 ## 5.1   lShowTCPEndpoint

207 ### 5.1.1   Description

208 lShowTCPEndpoint is a reusable recipe for displaying an instance of CIM_TCPProtocolEndpoint. A
209 recipe is defined for reuse by the `show` and `create` verbs applied to CIM_TCPProtocolEndpoint.

210 ### 5.1.2   Preconditions

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

212 `#all` indicates whether the "-all" option was specified.

213 ### 5.1.3   Pseudo Code

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

225 # 6   Mappings

226 The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in
227 the *Telnet Service Profile*. Requirements specified here related to support for a CLP verb for a particular
228 class are solely within the context of this profile.

229 ## 6.1   CIM_BindsTo

230 The `cd` and `help` verbs shall be supported as described in DSP0216.

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

236   **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	

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

6.1.1   Ordering of Results

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

242 • Results for CIM_BindsTo are unordered; therefore, no algorithm is defined.

6.1.2   Show

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

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

6.1.2.1   Show Multiple Instances – CIM_IPProtocolEndpoint

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

6.1.2.1.1   Command Form

252 **`show <CIM_BindsTo multiple instances>`**

6.1.2.1.2   CIM Requirements

254 See CIM_BindsTo in the "CIM Elements" section of the *Telnet Service Profile* for the list of mandatory
255 properties.

6.1.2.1.3   Behavior Requirements

6.1.2.1.3.1   Preconditions

258 `$instance` contains the instance of CIM_IPProtcolEndpoint which is referenced by CIM_BindsTo.

6.1.2.1.3.2   Pseudo Code

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

6.1.2.2   Show Multiple Instances – CIM_TCPProtocolEndpoint

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

266 **6.1.2.2.1 Command Form**

267 `show <CIM_BindsTo multiple instances>`

268 **6.1.2.2.2 CIM Requirements**

269 See CIM_BindsTo in the "CIM Elements" section of the *Telnet Service Profile* for the list of mandatory
270 properties.

271 **6.1.2.2.3 Behavior Requirements**

272 **6.1.2.2.3.1 Preconditions**

273 `$instance` contains the instance of CIM_TCPProtcolEndpoint which is referenced by CIM_BindsTo.

274 **6.1.2.2.3.2 Pseudo Code**

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

277 **6.1.2.3 Show a Single Instance – CIM_TelnetProtocolEndpoint Reference**

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

282 **6.1.2.3.1 Command Form**

283 `show <CIM_BindsTo single instance>`

284 **6.1.2.3.2 CIM Requirements**

285 See CIM_BindsTo in the "CIM Elements" section of the *Telnet Service Profile* for the list of mandatory
286 properties.

287 **6.1.2.3.3 Behavior Requirements**

288 **6.1.2.3.3.1 Preconditions**

289 `$instance` contains the instance of CIM_TelnetProtocolEndpoint which is referenced by CIM_BindsTo.

290 **6.1.2.3.3.2 Pseudo Code**

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

293 **6.1.2.4 Show a Single Instance – Both References A**

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

298 **6.1.2.4.1 Command Form**

299 `show <CIM_BindsTo single instance>`

300 **6.1.2.4.2	CIM Requirements**

301 See CIM_BindsTo in the "CIM Elements" section of the *Telnet Service Profile* for the list of mandatory
302 properties.

303 **6.1.2.4.3	Behavior Requirements**

304 **6.1.2.4.3.1	Preconditions**

305 `$instanceA` contains the instance of CIM_TelnetProtocolEndpoint which is referenced by CIM_BindsTo.

306 `$instanceB` contains the instance of CIM_TCPProtocolEndpoint which is referenced by CIM_BindsTo.

307 **6.1.2.4.3.2	Pseudo Code**

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

311 **6.1.2.4.4	Show a Single Instance – Both References B**

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

316 **6.1.2.4.5	Command Form**

317 **show <CIM_BindsTo *single instance*>**

318 **6.1.2.4.6	CIM Requirements**

319 See CIM_BindsTo in the "CIM Elements" section of the *Telnet Service Profile* for the list of mandatory
320 properties.

321 **6.1.2.4.7	Behavior Requirements**

322 **6.1.2.4.7.1	Preconditions**

323 `$instanceA` contains the instance of CIM_IPProtocolEndpoint which is referenced by CIM_BindsTo.

324 `$instanceB` contains the instance of CIM_TCPProtocolEndpoint which is referenced by CIM_BindsTo.

325 **6.1.2.4.7.2	Pseudo Code**

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

329 ## 6.2	CIM_ElementCapabilities

330 The `cd` and `help` verbs shall be supported as described in DSP0216.

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

336 **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	

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

6.2.1 Ordering of Results

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

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

6.2.2 Show

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

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

6.2.2.1 Show a Single Instance – CIM_TelnetCapabilities Reference

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

6.2.2.1.1 Command Form

355 `show <CIM_ElementCapabilities single instance>`

6.2.2.1.2 CIM Requirements

357 See CIM_ElementCapabilities in the "CIM Elements" section of the *Telnet Service Profile* for the list of
358 mandatory properties.

359 **6.2.2.1.3   Behavior Requirements**

360 **6.2.2.1.3.1   Preconditions**

361 $instance contains the instance of CIM_TelnetCapabilities which is referenced by
362 CIM_ElementCapabilities.

363 **6.2.2.1.3.2   Pseudo Code**

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

366 **6.2.2.2   Show a Single Instance – CIM_ProtocolService Reference**

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

371 **6.2.2.2.1   Command Form**

372 `show <CIM_ElementCapabilities single instance>`

373 **6.2.2.2.2   CIM Requirements**

374 See CIM_ElementCapabilities in the "CIM Elements" section of the *Telnet Service Profile* for the list of
375 mandatory properties.

376 **6.2.2.2.3   Behavior Requirements**

377 **6.2.2.2.3.1   Preconditions**

378 $instance contains the instance of CIM_ProtocolService which is referenced by
379 CIM_ElementCapabilities.

380 **6.2.2.2.3.2   Pseudo Code**

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

383 **6.2.2.3   Show a Single Instance – Both References**

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

387 **6.2.2.3.1   Command Form**

388 `show <CIM_ElementCapabilities single instance>`

389 **6.2.2.3.2   CIM Requirements**

390 See CIM_ElementCapabilities in the "CIM Elements" section of the *Telnet Service Profile* for the list of
391 mandatory properties.

392 **6.2.2.3.3 Behavior Requirements**

393 **6.2.2.3.3.1 Preconditions**

394 $instanceA contains the instance of CIM_TelnetCapabilities which is referenced by
395 CIM_ElementCapabilities.

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

398 **6.2.2.3.3.2 Pseudo Code**

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

402 **6.3 CIM_ElementSettingData**

403 The cd and help verbs shall be supported as described in DSP0216.

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

409 **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	

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

412 **6.3.1 Ordering of Results**

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

415 • Results for CIM_ElementSettingData are unordered; therefore, no algorithm is defined.

416 **6.3.2   Set**

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

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

421 **6.3.2.1   Set of IsNext**

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

424 **6.3.2.1.1   Command Form**

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

426 **6.3.2.1.2   CIM Requirements**

427 See CIM_ElementSettingData in the "CIM Elements" section of the *Telnet Service Profile* for the
428 CIM_ElementSettingData.IsNext property.

429 **6.3.2.1.3   Behavior Requirements**

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

435 **6.3.3   Show**

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

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

441 **6.3.3.1   Show Multiple Instances – CIM_TelnetSettingData and CIM_TelnetProtocolEndpoint**

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

446 **6.3.3.1.1   Command Form**

447 `show <CIM_ElementSettingData multiple instances>`

448 **6.3.3.1.2   CIM Requirements**

449 See CIM_ElementSettingData in the "CIM Elements" section of the *Telnet Service Profile* for the list of
450 mandatory properties.

451   **6.3.3.1.3   Behavior Requirements**

452   **6.3.3.1.3.1   Preconditions**

453   `$instance` contains the instance of CIM_TelnetSettingData which is referenced by
454   CIM_ElementSettingData.

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

456   **6.3.3.1.3.2   Pseudo Code**

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

465   **6.3.3.2   Show Multiple Instances – CIM_TelnetProtocolEndpoint Reference**

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

470   **6.3.3.2.1   Command Form**

471   **show <CIM_ElementSettingData *multiple instances*>**

472   **6.3.3.2.2   CIM Requirements**

473   See CIM_ElementSettingData in the "CIM Elements" section of the *[Telnet Service Profile](#)* for the list of
474   mandatory properties.

475   **6.3.3.2.3   Behavior Requirements**

476   **6.3.3.2.3.1   Preconditions**

477   `$instance` contains the instance of CIM_TelnetProtocolEndpoint which is referenced by
478   CIM_ElementSettingData.

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

480   **6.3.3.2.3.2   Pseudo Code**

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

489 **6.3.3.3   Show a Single Instance – CIM_TelnetSettingData and CIM_TelnetProtocolEndpoint**

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

493 **6.3.3.3.1   Command Form**

494 `show <CIM_ElementSettingData single instance>`

495 **6.3.3.3.2   CIM Requirements**

496 See CIM_ElementSettingData in the "CIM Elements" section of the *Telnet Service Profile* for the list of
497 mandatory properties.

498 **6.3.3.3.3   Behavior Requirements**

499 **6.3.3.3.3.1   Preconditions**

500 `$instanceA` contains the instance of CIM_TelnetSettingData which is referenced by
501 CIM_ElementSettingData.

502 `$instanceB` contains the instance of CIM_TelnetProtocolEndpoint which is referenced by
503 CIM_ElementSettingData.

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

505 **6.3.3.3.3.2   Pseudo Code**

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

514 **6.3.3.4   Show Multiple Instances – CIM_TelnetSettingData and CIM_ProtocolService**

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

519 **6.3.3.4.1   Command Form**

520 `show <CIM_ElementSettingData multiple instances>`

521 **6.3.3.4.2   CIM Requirements**

522 See CIM_ElementSettingData in the "CIM Elements" section of the *Telnet Service Profile* for the list of
523 mandatory properties.

524 **6.3.3.4.3   Behavior Requirements**

525 **6.3.3.4.3.1   Preconditions**

526 $instance contains the instance of CIM_TelnetSettingData which is referenced by
527 CIM_ElementSettingData.

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

529 **6.3.3.4.3.2   Pseudo Code**

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

538 **6.3.3.5   Show Multiple Instances – CIM_ProtocolService Reference**

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

543 **6.3.3.5.1   Command Form**

544 **show <CIM_ElementSettingData *multiple instances*>**

545 **6.3.3.5.2   CIM Requirements**

546 See CIM_ElementSettingData in the "CIM Elements" section of the *Telnet Service Profile* for the list of
547 mandatory properties.

548 **6.3.3.5.3   Behavior Requirements**

549 **6.3.3.5.3.1   Preconditions**

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

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

553 **6.3.3.5.3.2   Pseudo Code**

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

562 **6.3.3.6   Show a Single Instance – SettingData and ProtocolService**

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

566 **6.3.3.6.1   Command Form**

567 `show <CIM_ElementSettingData single instance>`

568 **6.3.3.6.2   CIM Requirements**

569 See CIM_ElementSettingData in the "CIM Elements" section of the *Telnet Service Profile* for the list of
570 mandatory properties.

571 **6.3.3.6.3   Behavior Requirements**

572 **6.3.3.6.3.1   Preconditions**

573 `$instanceA` contains the instance of CIM_TelnetSettingData which is referenced by
574 CIM_ElementSettingData.

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

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

578 **6.3.3.6.3.2   Pseudo Code**

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

587 ## 6.4   CIM_HostedAccessPoint

588 The `cd` and `help` verbs shall be supported as described in DSP0216.

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

594 **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	

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

6.4.1 Ordering of Results

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

600 • Results for CIM_HostedAccessPoint are unordered; therefore, no algorithm is defined.

6.4.2 Show

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

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

6.4.2.1 Show Multiple Instances

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

6.4.2.1.1 Command Form

612 `show <CIM_HostedAccessPoint multiple instances>`

6.4.2.1.2 CIM Requirements

614 See CIM_HostedAccessPoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of
615 mandatory properties.

6.4.2.1.3 Behavior Requirements

6.4.2.1.3.1 Preconditions

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

620 **6.4.2.1.3.2   Pseudo Code**

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

623 **6.4.2.2   Show a Single Instance – CIM_TCPProtocolEndpoint or CIM_TelnetProtocolEndpoint**
624 **Reference**

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

629 **6.4.2.2.1   Command Form**

```
630  show <CIM_HostedAccessPoint single instance>
```

631 **6.4.2.2.2   CIM Requirements**

632 See CIM_HostedAccessPoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of
633 mandatory properties.

634 **6.4.2.2.3   Behavior Requirements**

635 **6.4.2.2.3.1   Preconditions**

636 `$instance` contains the instance of CIM_TCPProtocolEndpoint or CIM_TelnetProtocolEndpoint which is
637 referenced by CIM_HostedAccessPoint.

638 **6.4.2.2.3.2   Pseudo Code**

```
639  &smShowAssociationInstances ( "CIM_HostedAccessPoint", $instance.getObjectPath() );
640  &smEnd;
```

641 **6.4.2.3   Show a Single Instance – Both References**

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

645 **6.4.2.3.1   Command Form**

```
646  show <CIM_HostedAccessPoint single instance>
```

647 **6.4.2.3.2   CIM Requirements**

648 See CIM_HostedAccessPoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of
649 mandatory properties.

650 **6.4.2.3.3   Behavior Requirements**

651 **6.4.2.3.3.1   Preconditions**

652 `$instanceA` contains the instance of CIM_ComputerSystem which is referenced by
653 CIM_HostedAccessPoint.

654 `$instanceB` contains the instance of CIM_TelnetProtocolEndpoint or CIM_TCPProtocolEndpoint which
655 is referenced by CIM_HostedAccessPoint.

656 **6.4.2.3.3.2 Pseudo Code**

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

660 ## 6.5 CIM_HostedService

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

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

667 **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	

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

670 ### 6.5.1 Ordering of Results

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

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

674 ### 6.5.2 Show

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

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

678 **6.5.2.1 Show Multiple Instances**

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

682 **6.5.2.1.1   Command Form**

683 `show <CIM_HostedService multiple instances>`

684 **6.5.2.1.2   CIM Requirements**

685 See CIM_HostedService in the "CIM Elements" section of the *Telnet Service Profile* for the list of
686 mandatory properties.

687 **6.5.2.1.3   Behavior Requirements**

688 **6.5.2.1.3.1   Preconditions**

689 `$instance` contains the instance of CIM_ComputerSystem which is referenced by CIM_HostedService.

690 **6.5.2.1.3.2   Pseudo Code**

691 `&smShowAssociationInstances ( "CIM_HostedService", $instance.getObjectPath() );`
692 `&smEnd;`

693 **6.5.2.2   Show a Single Instance – CIM_ProtocolService Reference**

694 This command form is for the `show` verb applied to a single instance. This command form corresponds to
695 a `show` command issued against CIM_HostedService where the reference specified is to an instance of
696 CIM_ProtocolService. An instance of CIM_ProtocolService is referenced by exactly one instance of
697 CIM_HostedService. Therefore, a single instance will be returned.

698 **6.5.2.2.1   Command Form**

699 `show <CIM_HostedService single instance>`

700 **6.5.2.2.2   CIM Requirements**

701 See CIM_HostedService in the "CIM Elements" section of the *Telnet Service Profile* for the list of
702 mandatory properties.

703 **6.5.2.2.3   Behavior Requirements**

704 **6.5.2.2.3.1   Preconditions**

705 `$instance` contains the instance of CIM_ProtocolService which is referenced by CIM_HostedService.

706 **6.5.2.2.3.2   Pseudo Code**

707 `&smShowAssociationInstances ( "CIM_HostedService", $instance.getObjectPath() );`
708 `&smEnd;`

709 **6.5.2.3   Show a Single Instance – Both References**

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

713 **6.5.2.3.1   Command Form**

714 `show <CIM_HostedService single instance>`

715 **6.5.2.3.2   CIM Requirements**

716 See CIM_HostedService in the "CIM Elements" section of the *Telnet Service Profile* for the list of
717 mandatory properties.

718 **6.5.2.3.3   Behavior Requirements**

719 **6.5.2.3.3.1   Preconditions**

720 $instanceA contains the instance of CIM_ComputerSystem which is referenced by
721 CIM_HostedService.

722 $instanceB contains the instance of CIM_ProtocolService which is referenced by CIM_HostedService.

723 **6.5.2.3.3.2   Pseudo Code**

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

727 ## 6.6   CIM_ProvidesEndpoint

728 The cd and help verbs shall be supported as described in DSP0216.

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

734 **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	

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

737 **6.6.1   Ordering of Results**

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

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

6.6.2   Show

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

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

6.6.2.1   Show a Single Instance – CIM_TelnetProtocolEndpoint Reference

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

6.6.2.1.1   Command Form

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

6.6.2.1.2   CIM Requirements

See CIM_ProvidesEndpoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of mandatory properties.

6.6.2.1.3   Behavior Requirements

6.6.2.1.3.1   Preconditions

`$instance` contains the instance of CIM_TelnetProtocolEndpoint which is referenced by CIM_ProvidesEndpoint.

6.6.2.1.3.2   Pseudo Code

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

6.6.2.2   Show Multiple Instances – CIM_ProtocolService Reference

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

6.6.2.2.1   Command Form

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

6.6.2.2.2   CIM Requirements

See CIM_ProvidesEndpoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of mandatory properties.

6.6.2.2.3   Behavior Requirements

6.6.2.2.3.1   Preconditions

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

778 **6.6.2.2.3.2 Pseudo Code**

```
779   &smShowAssociationInstances ( "CIM_ProvidesEndpoint", $instance.getObjectPath() );
780   &smEnd;
```

781 **6.6.2.3 Show a Single Instance – Both References**

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

785 **6.6.2.3.1 Command Form**

786 **show <CIM_ProvidesEndpoint *single instance*>**

787 **6.6.2.3.2 CIM Requirements**

788 See CIM_ProvidesEndpoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of
789 mandatory properties.

790 **6.6.2.3.3 Behavior Requirements**

791 **6.6.2.3.3.1 Preconditions**

792 `$instanceA` contains the instance of CIM_TelnetProtocolEndpoint which is referenced by
793 CIM_ProvidesEndpoint.

794 `$instanceB` contains the instance of CIM_ProtocolService which is referenced by
795 CIM_ProvidesEndpoint.

796 **6.6.2.3.3.2 Pseudo Code**

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

800 ## 6.7 CIM_ProtocolService

801 The `cd` and `help` verbs shall be supported as described in DSP0216.

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

807 **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 6.7.4.2.4.
stop	May	See 6.7.5.

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

6.7.1   Ordering of Results

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

812   • Results for CIM_ProtocolService are unordered; therefore, no algorithm is defined.

6.7.2   Reset

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

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

6.7.2.1   Reset a Single Instance

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

6.7.2.1.1   Command Form

823   **`reset <CIM_ProtocolService single instance>`**

6.7.2.1.2   CIM Requirements

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

6.7.2.1.3   Behavior Requirements

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

6.7.3   Set

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

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

839 **6.7.3.1 General Usage of Set for a Single Property**

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

842 The requirement for supporting modification of a property using this command form shall be equivalent to
843 the requirement for supporting modification of the property using the ModifyInstance operation as defined
844 in the *Telnet Service Profile*.

845 **6.7.3.1.1 Command Form**

846 ```
set <CIM_ProtocolService single instance> <propertyname>=<propertyvalue>
```

847 **6.7.3.1.2 CIM Requirements**

848 See CIM_ProtocolService in the "CIM Elements" section of the *Telnet Service Profile* for the list of
849 modifiable properties.

850 **6.7.3.1.3 Behavior Requirements**

851 ```
$instance=<CIM_ProtocolService single instance>
```
852 ```
#propertyNames[] = {<propertyname>};
```
853 ```
#propertyValues[] = {<propertyvalue>};
```
854 ```
&smSetInstance ($instance, #propertyNames[], #propertyValues[]);
```
855 ```
&smEnd;
```

856 **6.7.3.2 General Usage of Set for Multiple Properties**

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

860 The requirement for supporting modification of a property using this command form shall be equivalent to
861 the requirement for supporting modification of the property using the ModifyInstance operation as defined
862 in the *Telnet Service Profile*.

863 **6.7.3.2.1 Command Form**

864 ```
set <CIM_ProtocolService single instance> <propertyname1>=<propertyvalue1>
```
865 ```
    <propertynamen>=<propertyvaluen>
```

866 **6.7.3.2.2 CIM Requirements**

867 See CIM_ProtocolService in the "CIM Elements" section of the *Telnet Service Profile* for the list of
868 mandatory properties.

869 **6.7.3.2.3 Behavior Requirements**

870 ```
$instance=<CIM_ProtocolService single instance>
```
871 ```
#propertyNames[] = {<propertyname>};
```
872 ```
for #i < n
```
873 ```
    {
```
874 ```
    #propertyNames[#i] = <propertname#i>
```
875 ```
    #propertyValues[#i] = <propertyvalue#i>
```
876 ```
    }
```
877 ```
&smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
```
878 ```
&smEnd;
```

879    **6.7.4    Show**

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

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

883    **6.7.4.1    Show a Single Instance**

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

885    **6.7.4.1.1    Command Form**

886    `show <CIM_ProtocolService single instance>`

887    **6.7.4.1.2    CIM Requirements**

888    See CIM_ProtocolService in the "CIM Elements" section of the *Telnet Service Profile* for the list of
889    mandatory properties.

890    **6.7.4.1.3    Behavior Requirements**

891    **6.7.4.1.3.1      Preconditions**

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

893    **6.7.4.1.3.2      Pseudo Code**

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

902   **6.7.4.2   Show Multiple Instances**

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

905   **6.7.4.2.1   Command Form**

906   `show <CIM_ProtocolService multiple instances>`

907   **6.7.4.2.2   CIM Requirements**

908   See CIM_ProtocolService in the "CIM Elements" section of the *Telnet Service Profile* for the list of
909   mandatory properties.

910   **6.7.4.2.3   Behavior Requirements**

911   **6.7.4.2.3.1   Preconditions**

912   `$containerInstance` contains the instance of CIM_ComputerSystem for which scoped instances of
913   the CIM_ProtocolService are displayed. The *Telnet Service Profile* requires that the CIM_ProtocolService
914   instance be associated with its scoping system via an instance of the CIM_HostedService association.

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

916 **6.7.4.2.3.2   Pseudo Code**

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

925 **6.7.4.2.4   Start**

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

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

930 **6.7.4.3   Start a Single Instance**

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

932 **6.7.4.3.1   Command Form**

933 **start <CIM_ProtocolService *single instance*>**

934 **6.7.4.3.2   CIM Requirements**

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

941 **6.7.4.3.3   Behavior Requirements**

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

945 **6.7.5   Stop**

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

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

949 **6.7.5.1   Stop a Single Instance**

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

951 **6.7.5.1.1   Command Form**

952 `stop <CIM_ProtocolService single instance>`

953 **6.7.5.1.2   CIM Requirements**

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

960 **6.7.5.1.3   Behavior Requirements**

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

964 ## 6.8   CIM_ServiceAccessBySAP

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

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

971   **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	

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

974 **6.8.1   Ordering of Results**

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

977   •   Results for CIM_ServiceAccessBySAP are unordered; therefore, no algorithm is defined.

978 **6.8.2 Show**

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

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

984 **6.8.2.1 Show Multiple Instances – CIM_ProtocolService Reference**

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

988 **6.8.2.1.1 Command Form**

989 `show <CIM_ServiceAccessBySAP multiple instances>`

990 **6.8.2.1.2 CIM Requirements**

991 See CIM_ProtocolService in the "CIM Elements" section of the *Telnet Service Profile* for the list of
992 mandatory properties.

993 **6.8.2.1.3 Behavior Requirements**

994 **6.8.2.1.3.1 Preconditions**

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

997 **6.8.2.1.3.2 Pseudo Code**

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

1000 **6.8.2.2 Show Multiple Instances – CIM_TCPProtocolEndpoint Reference**

1001 This command form is for the `show` verb applied to multiple instances. This command form corresponds
1002 to a `show` command issued against CIM_ServiceAccessBySAP where the reference specified is to an
1003 instance of CIM_TCPProtocolEndpoint.

1004 **6.8.2.2.1 Command Form**

1005 `show <CIM_ServiceAccessBySAP multiple instances>`

1006 **6.8.2.2.2 CIM Requirements**

1007 See CIM_ServiceAccessBySAP in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1008 mandatory properties.

1009 **6.8.2.2.3 Behavior Requirements**

1010 **6.8.2.2.3.1 Preconditions**

1011 `$instance` contains the instance of CIM_TCPProtocolEndpoint which is referenced by
1012 CIM_ServiceAccessBySAP.

1013 **6.8.2.2.3.2   Pseudo Code**

```
1014  &smShowAssociationInstances ( "CIM_ServiceAccessBySAP", $instance.getObjectPath() );
1015  &smEnd;
```

1016 **6.8.2.3   Show a Single Instance – Both References**

1017 This command form is for the `show` verb applied to a single instance. This command form corresponds to
1018 a `show` command issued against CIM_ServiceAccessBySAP where both references are specified and
1019 therefore the desired instance is unambiguously identified.

1020 **6.8.2.3.1   Command Form**

```
1021  show <CIM_ServiceAccessBySAP single instance>
```

1022 **6.8.2.3.2   CIM Requirements**

1023 See CIM_ServiceAccessBySAP in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1024 mandatory properties.

1025 **6.8.2.3.3   Behavior Requirements**

1026 **6.8.2.3.3.1   Preconditions**

1027 `$instanceA` contains the instance of CIM_TCPProtocolEndpoint which is referenced by
1028 CIM_ServiceAccessBySAP.

1029 `$instanceB` contains the instance of CIM_ProtocolService which is referenced by
1030 CIM_ServiceAccessBySAP.

1031 **6.8.2.3.3.2   Pseudo Code**

```
1032  &smShowAssociationInstance ( "CIM_ServiceAccessBySAP", $instanceA.getObjectPath(),
1033     $instanceB.getObjectPath() );
1034  &smEnd;
```

1035 **6.9   CIM_TelnetCapabilities**

1036 The `cd` and `help` verbs shall be supported as described in DSP0216.

1037 Table 10 lists each SM CLP verb, the required level of support for the verb in conjunction with instances
1038 of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the
1039 verb and target. Table 10 is for informational purposes only; in case of a conflict between Table 10 and
1040 requirements detailed in the following sections, the text detailed in the following sections supersedes the
1041 information in Table 10.

1042 **Table 10 – Command Verb Requirements for CIM_TelnetCapabilities**

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 6.9.2.
start	Not supported	
stop	Not supported	

1043 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,
1044 `reset`, `set`, `start`, and `stop`.

6.9.1 Ordering of Results

1045

1046 When results are returned for multiple instances of CIM_TelnetCapabilities, implementations shall utilize
1047 the following algorithm to produce the natural (that is, default) ordering:

1048 - Results for CIM_TelnetCapabilities are unordered; therefore, no algorithm is defined.

6.9.2 Show

1049

1050 This section describes how to implement the `show` verb when applied to an instance of
1051 CIM_TelnetCapabilities. Implementations shall support the use of the `show` verb with
1052 CIM_TelnetCapabilities.

1053 The `show` verb is used to display information about an instance or instances of the
1054 CIM_TelnetCapabilities class.

6.9.2.1 Show a Single Instance

1055

1056 This command form is for the `show` verb applied to a single instance of CIM_TelnetCapabilities.

6.9.2.1.1 Command Form

1057

1058 `show <CIM_TelnetCapabilities single instance>`

6.9.2.1.2 CIM Requirements

1059

1060 See CIM_TelnetCapabilities in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1061 mandatory properties.

6.9.2.1.3 Behavior Requirements

1062

6.9.2.1.3.1 Preconditions

1063

1064 `#all` is true if the "-all" option was specified with the command; otherwise, `#all` is false.

1065 **6.9.2.1.3.2   Pseudo Code**

```
1066  $instance=<CIM_TelnetCapabilities single instance>
1067  #propertylist[] = NULL;
1068  if ( false == #all)
1069      {
1070      #propertylist[] = {//all mandatory non-key properties }
1071      }
1072  &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
1073  &smEnd;
```

1074 **6.9.2.2   Show Multiple Instances**

1075 This command form is for the `show` verb applied to multiple instances of CIM_TelnetCapabilities. This
1076 command form corresponds to UFsT-based selection within a capabilities collection.

1077 **6.9.2.2.1   Command Form**

```
1078  show <CIM_TelnetCapabilities multiple instances>
```

1079 **6.9.2.2.2   CIM Requirements**

1080 See CIM_TelnetCapabilities in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1081 mandatory properties.

1082 **6.9.2.2.3   Behavior Requirements**

1083 **6.9.2.2.3.1   Preconditions**

1084 `$containerInstance` contains the instance of CIM_ConcreteCollection for which contained
1085 CIM_Capabilities instances are displayed. CIM_Capabilities instances are addressed via an aggregating
1086 instance of CIM_ConcreteCollection.

1087 `#all` is true if the "-all" option was specified with the command; otherwise, `#all` is false.

1088 **6.9.2.2.3.2   Pseudo Code**

```
1089  #propertylist[] = NULL;
1090  if ( false == #all)
1091      {
1092      #propertylist[] = {//all mandatory non-key properties }
1093      }
1094  &smShowInstances ( "CIM_TelnetCapabilities", "CIM_MemberOfCollection",
1095      $containerInstance.getObjectPath(), #propertylist[] );
1096  &smEnd;
```

1097 ## 6.10   CIM_TelnetProtocolEndpoint

1098 The `cd` and `help` verbs shall be supported as described in DSP0216.

1099 Table 11 lists each SM CLP verb, the required level of support for the verb in conjunction with instances
1100 of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the
1101 verb and target. Table 11 is for informational purposes only; in case of a conflict between Table 11 and
1102 requirements detailed in the following sections, the text detailed in the following sections supersedes the
1103 information in Table 11.

1104 **Table 11 – Command Verb Requirements for CIM_TelnetProtocolEndpoint**

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.
show	Shall	See 6.10.3.
start	Not supported	
stop	May	See 6.10.4.

1105 No mapping is defined for the following verbs for the specified target: `dump` and `load`.

6.10.1 Ordering of Results

1106

1107 When results are returned for multiple instances of CIM_TelnetProtocolEndpoint, implementations shall
1108 utilize the following algorithm to produce the natural (that is, default) ordering:

1109 • Results for CIM_TelnetProtocolEndpoint are unordered; therefore, no algorithm is defined.

6.10.2 Set

1110

1111 This section describes how to implement the `set` verb when it is applied to an instance of
1112 CIM_TelnetProtocolEndpoint. Implementations may support the use of the `set` verb with
1113 CIM_TelnetProtocolEndpoint.

1114 The `set` verb is used to modify descriptive properties of the CIM_TelnetProtocolEndpoint instance.

6.10.2.1 General Usage of Set for a Single Property

1115

1116 This command form corresponds to the general usage of the `set` verb to modify a single property of a
1117 target instance. This is the most common case.

1118 The requirement for supporting modification of a property using this command form shall be equivalent to
1119 the requirement for supporting modification of the property using the ModifyInstance operation as defined
1120 in the *Telnet Service Profile*.

6.10.2.1.1 Command Form

1121

1122 `set <CIM_TelnetProtocolEndpoint single instance> <propertyname>=<propertyvalue>`

6.10.2.1.2 CIM Requirements

1123

1124 See CIM_TelnetProtocolEndpoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1125 modifiable properties.

6.10.2.1.3 Behavior Requirements

1126

1127 `$instance=<CIM_TelnetProtocolEndpoint single instance>`
1128 `#propertyNames[] = {<propertyname>};`
1129 `#propertyValues[] = {<propertyvalue>};`

```
1130   &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
1131   &smEnd;
```

6.10.2.2 General Usage of Set for Multiple Properties

1133 This command form corresponds to the general usage of the `set` verb to modify multiple properties of a
1134 target instance where there is not an explicit relationship between the properties. This is the most
1135 common case.

1136 The requirement for supporting modification of a property using this command form shall be equivalent to
1137 the requirement for supporting modification of the property using the ModifyInstance operation as defined
1138 in the *Telnet Service Profile*.

6.10.2.2.1 Command Form

```
1140   set <CIM_TelnetProtocolEndpoint single instance> <propertyname1>=<propertyvalue1>
1141       <propertynamen>=<propertyvaluen>
```

6.10.2.2.2 CIM Requirements

1143 See CIM_TelnetProtocolEndpoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1144 mandatory properties.

6.10.2.2.3 Behavior Requirements

```
1146   $instance=<CIM_TelnetProtocolEndpoint single instance>
1147   #propertyNames[] = {<propertyname>};
1148   for #i < n
1149       {
1150       #propertyNames[#i] = <propertname#i>
1151       #propertyValues[#i] = <propertyvalue#i>
1152       }
1153   &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
1154   &smEnd;
```

6.10.3 Show

1156 This section describes how to implement the `show` verb when applied to an instance of
1157 CIM_TelnetProtocolEndpoint. Implementations shall support the use of the `show` verb with
1158 CIM_TelnetProtocolEndpoint.

1159 The `show` verb is used to display information about a Telnet session.

1160 Note that CIM_BindsTo and CIM_HostedAccessPoint are both Addressing Associations. Thus, an
1161 implementation of the SM CLP has a choice when exposing the address for an instance of
1162 CIM_TelnetProtocolEndpoint. For completeness, mappings are shown for both associations, though only
1163 one would be applicable in a given implementation.

6.10.3.1 Show a Single Instance

1165 This command form is for the `show` verb applied to a single instance of CIM_TelnetProtocolEndpoint.

6.10.3.1.1 Command Form

```
1167   show <CIM_TelnetProtocolEndpoint single instance>
```

1168 **6.10.3.1.2   CIM Requirements**

1169 See CIM_TelnetProtocolEndpoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1170 mandatory properties.

1171 **6.10.3.1.3   Behavior Requirements**

1172 **6.10.3.1.3.1   Preconditions**

1173 #all is true if the "-all" option was specified with the command; otherwise, #all is false.

1174 **6.10.3.1.3.2   Pseudo Code**

```
1175  $instance=<CIM_TelnetProtocolEndpoint single instance>
1176  #propertylist[] = NULL;
1177  if ( false == #all)
1178      {
1179      #propertylist[] = {//all mandatory non-key properties }
1180      }
1181  &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
1182  &smEnd;
```

1183 **6.10.3.2   Show Multiple Instances Scoped by a System**

1184 This command form is for the show verb applied to multiple instances of CIM_TelnetProtocolEndpoint.
1185 This command form corresponds to UFsT-based selection within a scoping system.

1186 **6.10.3.2.1   Command Form**

```
1187  show <CIM_TelnetProtocolEndpoint multiple instances>
```

1188 **6.10.3.2.2   CIM Requirements**

1189 See CIM_TelnetProtocolEndpoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1190 mandatory properties.

1191 **6.10.3.2.3   Behavior Requirements**

1192 **6.10.3.2.3.1   Preconditions**

1193 $containerInstance contains the instance of CIM_ComputerSystem for which scoped endpoints
1194 (CIM_TelnetProtocolEndpoint instances) are displayed. The *Telnet Service Profile* requires that the
1195 CIM_TelnetProtocolEndpoint instance be associated with its scoping system via an instance of the
1196 CIM_HostedAccessPoint association.

1197 #all is true if the "-all" option was specified with the command; otherwise, #all is false.

1198 **6.10.3.2.3.2   Pseudo Code**

```
1199  #propertylist[] = NULL;
1200  if (false == #all)
1201      {
1202      #propertylist[] = {//all mandatory non-key properties }
1203      }
1204  &smShowInstances ( "CIM_TelnetProtocolEndpoint", "CIM_HostedAccessPoint",
1205      $containerInstance.getObjectPath(), #propertylist[] );
1206  &smEnd;
```

1207 **6.10.3.3   Show Multiple Instances Scoped by a TCPProtocolEndpoint**

1208 This command form is for the `show` verb applied to multiple instances of CIM_TelnetProtocolEndpoint.
1209 This command form corresponds to UFsT-based selection within a scoping CIM_TCPProtocolEndpoint
1210 instance.

1211 **6.10.3.3.1   Command Form**

1212 ```
show <CIM_TelnetProtocolEndpoint multiple instances>
```

1213 **6.10.3.3.2   CIM Requirements**

1214 See CIM_TelnetProtocolEndpoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1215 mandatory properties.

1216 **6.10.3.3.3   Behavior Requirements**

1217 **6.10.3.3.3.1   Preconditions**

1218 `$containerInstance` contains the instance of CIM_TCPProtocolEndpoint for which scoped endpoints
1219 (CIM_TelnetProtocolEndpoint instances) are displayed. The *Telnet Service Profile* requires that the
1220 CIM_TelnetProtocolEndpoint instance be associated with a CIM_TCPProtocolEndpoint instance via an
1221 instance of the CIM_BindsTo association.

1222 `#all` is true if the "-all" option was specified with the command; otherwise, `#all` is false.

1223 **6.10.3.3.3.2   Pseudo Code**

1224 ```
#propertylist[] = NULL;
```
1225 ```
if (false == #all)
```
1226 ```
    {
```
1227 ```
    #propertylist[] = {//all mandatory non-key properties };
```
1228 ```
    }
```
1229 ```
&smShowInstances ( "CIM_TelnetProtocolEndpoint", "CIM_BindsTo",
```
1230 ```
    $containerInstance.getObjectPath(), #propertylist[]);
```
1231 ```
&smEnd;
```

1232 **6.10.4   Stop**

1233 This section describes how to implement the `stop` verb when applied to an instance of
1234 CIM_TelnetProtocolEndpoint. Implementations may support the use of the `stop` verb with
1235 CIM_TelnetProtocolEndpoint.

1236 The `stop` verb is used to disable an endpoint.

1237 **6.10.4.1   Stop a Single Instance**

1238 This command form is for the `stop` verb applied to a single instance of CIM_TelnetProtocolEndpoint. The
1239 lifecycle of a Telnet session corresponds to the lifecycle of the CIM_TelnetProtocolEndpoint which
1240 represents it. Therefore, stopping a Telnet service corresponds to a delete of the underlying instance.

1241 **6.10.4.1.1   Command Form**

1242 ```
stop <CIM_TelnetProtocolEndpoint single instance>
```

1243 **6.10.4.1.2 CIM Requirements**

1244 See CIM_TelnetProtocolEndpoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1245 mandatory properties.

1246 **6.10.4.1.3 Behavior Requirements**

```
1247  $instance=<CIM_TelnetProtocolEndpoint single instance>
1248  &smDeleteInstance ( $instance.getObjectPath() );
1249  &smEnd;
```

1250 **6.11 CIM_TelnetSettingData**

1251 The `cd` and `help` verbs shall be supported as described in DSP0216.

1252 Table 12 lists each SM CLP verb, the required level of support for the verb in conjunction with instances
1253 of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the
1254 verb and target. Table 12 is for informational purposes only; in case of a conflict between Table 12 and
1255 requirements detailed in the following sections, the text detailed in the following sections supersedes the
1256 information in Table 12.

1257 **Table 12 – Command Verb Requirements for CIM_TelnetSettingData**

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	

1258 No mapping is defined for the following verbs for the specified target: `dump` and `load`.

1259 **6.11.1 Ordering of Results**

1260 When results are returned for multiple instances of CIM_TelnetSettingData, implementations shall utilize
1261 the following algorithm to produce the natural (that is, default) ordering:

1262 • Results for CIM_TelnetSettingData are unordered; therefore, no algorithm is defined.

1263 **6.11.2 Set**

1264 This section describes how to implement the `set` verb when it is applied to an instance of
1265 CIM_TelnetSettingData. Implementations may support the use of the `set` verb with
1266 CIM_TelnetSettingData.

1267 The `set` verb is used to modify configuration represented by an instance of CIM_TelnetSettingData.

1268 **6.11.2.1 General Usage of Set for a Single Property**

1269 This command form corresponds to the general usage of the set verb to modify a single property of a
1270 target instance. This is the most common case.

1271 The requirement for supporting modification of a property using this command form shall be equivalent to
1272 the requirement for supporting modification of the property using the ModifyInstance operation as defined
1273 in the *Telnet Service Profile*.

1274 **6.11.2.1.1 Command Form**

1275 `set <CIM_TelnetSettingData single instance> <propertyname>=<propertyvalue>`

1276 **6.11.2.1.2 CIM Requirements**

1277 See CIM_TelnetSettingData in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1278 modifiable properties.

1279 **6.11.2.1.3 Behavior Requirements**

```
1280  $instance=<CIM_TelnetSettingData single instance>
1281  #propertyNames[] = {<propertyname>};
1282  #propertyValues[] = {<propertyvalue>};
1283  &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
1284  &smEnd;
```

1285 **6.11.2.2 General Usage of Set for Multiple Properties**

1286 This command form corresponds to the general usage of the set verb to modify multiple properties of a
1287 target instance where there is not an explicit relationship between the properties. This is the most
1288 common case.

1289 The requirement for supporting modification of a property using this command form shall be equivalent to
1290 the requirement for supporting modification of the property using the ModifyInstance operation as defined
1291 in the *Telnet Service Profile*.

1292 **6.11.2.2.1 Command Form**

```
1293  set <CIM_TelnetSettingData single instance> <propertyname1>=<propertyvalue1>
1294     <propertynamen>=<propertyvaluen>
```

1295 **6.11.2.2.2 CIM Requirements**

1296 See CIM_TelnetSettingData in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1297 mandatory properties.

1298 **6.11.2.2.3 Behavior Requirements**

```
1299  $instance=<CIM_TelnetSettingData single instance>
1300  #propertyNames[] = {<propertyname>};
1301  for #i < n
1302     {
1303     #propertyNames[#i] = <propertname#i>
1304     #propertyValues[#i] = <propertyvalue#i>
1305     }
1306  &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
1307  &smEnd;
```

1308 **6.11.3 Show**

1309 This section describes how to implement the show verb when applied to an instance of
1310 CIM_TelnetSettingData. Implementations shall support the use of the show verb with
1311 CIM_TelnetSettingData.

1312 The show verb is used to display information about the CIM_TelnetSettingData instance.

1313 **6.11.3.1 Show a Single Instance**

1314 This command form is for the show verb applied to a single instance of CIM_TelnetSettingData.

1315 **6.11.3.1.1 Command Form**

1316 `show <CIM_TelnetSettingData single instance>`

1317 **6.11.3.1.2 CIM Requirements**

1318 See CIM_TelnetSettingData in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1319 mandatory properties.

1320 **6.11.3.1.3 Behavior Requirements**

1321 **6.11.3.1.3.1 Preconditions**

1322 #all is true if the "-all" option was specified with the command; otherwise, #all is false.

1323 **6.11.3.1.3.2 Pseudo Code**

1324 `$instance=<CIM_TelnetSettingData single instance>`
1325 `&lShowTCPEndpoint ( $instance, #all );`
1326 `&smEnd;`

1327 **6.11.3.2 Show Multiple Instances Scoped by ConcreteCollection**

1328 This command form is for the show verb applied to multiple instances of CIM_TelnetSettingData. This
1329 command form corresponds to UFsT-based selection within an instance of CIM_ConcreteCollection.

1330 **6.11.3.2.1 Command Form**

1331 `show <CIM_TelnetSettingData multiple instances>`

1332 **6.11.3.2.2 CIM Requirements**

1333 See CIM_TelnetSettingData in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1334 mandatory properties.

1335 **6.11.3.2.3 Behavior Requirements**

1336 **6.11.3.2.3.1 Preconditions**

1337 $containerInstance contains the instance of CIM_ConcreteCollection for which contained
1338 CIM_TelnetSettingData instances are displayed. The *SMASH Collections Profile* requires that the
1339 CIM_TelnetSettingData instances be aggregated into an addressing collection via
1340 CIM_MemberOfCollection.

1341 #all is true if the "-all" option was specified with the command; otherwise, #all is false.

6.11.3.2.3.2 Pseudo Code

```
#propertylist[] = NULL;
//this property list will match the property list in lShowTCPEndpoint()
if (false == #all)
    {
    #propertylist[] = {//all mandatory non-key properties }
    }
&smShowInstances ( "CIM_TelnetSettingData", "CIM_MemberOfCollection",
    $containerInstance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.12 CIM_TCPProtocolEndpoint

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

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

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	

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

6.12.1 Ordering of Results

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

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

6.12.2 Create

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

1369 The `create` verb is used to create an additional CIM_TCPProtocolEndpoint instance representing a port
1370 upon which the Telnet service is listening.

1371 **6.12.2.1 Specifying the Required Port Number**

1372 In order to create an instance of CIM_TCPProtocolEndpoint, a client is required to supply the desired IP
1373 port.

1374 **6.12.2.1.1 Command Form**

1375 ```
create <CIM_TCPProtocolEndpoint single instance> portnumber=<desiredport>
```

1376 **6.12.2.1.2 CIM Requirements**

1377 See CIM_TCPProtocolEndpoint in the "CIM Elements" section of the *Telnet Service Profile* for the
1378 CIM_ProtocolService.AddListeningPort property.

1379 **6.12.2.1.3 Behavior Requirements**

1380 **6.12.2.1.3.1 Preconditions**

1381 `$Service` contains the CIM_ProtocolService instance for which we are creating a new endpoint.

1382 **6.12.2.1.3.2 Pseudo Code**

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

```
1412        //method invocation failed
1413        if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )    {
1414            // if the method invocation contains an embedded error
1415            // use it for the Error for the overall job
1416            &smAddError($job, #Error.$error[0]);
1417            &smMakeCommandStatus($job);
1418            &smEnd;
1419        }
1420        else if (#Error.code == 17)    {
1421        //trap for CIM_METHOD_NOT_FOUND
1422        //and make nice Unsupported msg.
1423            //unsupported
1424            $OperationError = smNewInstance("CIM_Error");
1425            //CIM_ERR_NOT_SUPPORTED
1426            $OperationError.CIMStatusCode = 7;
1427            //Other
1428            $OperationError.ErrorType = 1;
1429            //Low
1430            $OperationError.PerceivedSeverity = 2;
1431            $OperationError.OwningEntity = DMTF:SMCLP;
1432            $OperationError.MessageID = 0x00000001;
1433            $OperationError.Message = "Operation is not supported.";
1434            &smAddError($job, $OperationError);
1435            &smMakeCommandStatus($job);
1436            &smEnd;
1437        }
1438        else {
1439            //operation failed, but no detailed error instance, need to make one up
1440            //make an Error instance and associate with job for Operation
1441                $OperationError = smNewInstance("CIM_Error");
1442                //CIM_ERR_FAILED
1443                $OperationError.CIMStatusCode = 1;
1444                //Software Error
1445                $OperationError.ErrorType = 4;
1446                //Unknown
1447                $OperationError.PerceivedSeverity = 0;
1448                $OperationError.OwningEntity = DMTF:SMCLP;
1449                $OperationError.MessageID = 0x00000009;
1450                $OperationError.Message = "An internal software error has occurred.";
1451                &smAddError($job, $OperationError);
1452                &smMakeCommandStatus($job);
1453                &smEnd;
1454        }
1455    }//if CIM op failed
1456    else if (0 == #returnStatus) {
1457        //completed successfully
1458        &lShowTCPEndpoint($Endpoint, "false");
1459        &smEnd;
1460    }
```

```
1461      else if (1 == #returnStatus)   {
1462      //and make nice Unsupported msg.
1463         $OperationError = smNewInstance("CIM_Error");
1464         //CIM_ERR_NOT_SUPPORTED
1465         $OperationError.CIMStatusCode = 7;
1466         //Other
1467         $OperationError.ErrorType = 1;
1468         //Low
1469         $OperationError.PerceivedSeverity = 2;
1470         $OperationError.OwningEntity = DMTF:SMCLP;
1471         $OperationError.MessageID = 0x00000001;
1472         $OperationError.Message = "Operation is not supported.";
1473         &smAddError($job, $OperationError);
1474         &smMakeCommandStatus($job);
1475         &smEnd;
1476      }
1477 else   {
1478      //generic failure
1479      $OperationError = smNewInstance("CIM_Error");
1480      //CIM_ERR_FAILED
1481      $OperationError.CIMStatusCode = 1;
1482      //Other
1483      $OperationError.ErrorType = 1;
1484      //Low
1485      $OperationError.PerceivedSeverity = 2;
1486      $OperationError.OwningEntity = DMTF:SMCLP;
1487      $OperationError.MessageID = 0x00000002;
1488      $OperationError.Message = "Failed. No further information is available.";
1489      &smAddError($job, $OperationError);
1490      &smMakeCommandStatus($job);
1491 }
```

6.12.3 Delete

1492

1493 This section describes how to implement the delete verb when applied to an instance of
1494 CIM_TCPProtocolEndpoint. Implementations may support the use of the delete verb with
1495 CIM_TCPProtocolEndpoint.

1496 The delete command is used to remove an instance of CIM_TCPProtocolEndpoint which represents a
1497 virtual MAC.

6.12.3.1 Delete a Single Instance

1498

1499 Delete a single instance of CIM_TCPProtocolEndpoint.

6.12.3.1.1 Command Form

1500

1501 **delete <CIM_TCPProtocolEndpoint *single instance*>**

6.12.3.1.2 CIM Requirements

1502

1503 See CIM_TCPProtocolEndpoint in the "CIM Elements" section of the *Telnet Service Profile* for the
1504 CIM_TCPProtocolEndpoint property.

1505 **6.12.3.1.3   Behavior Requirements**

```
1506  $instance=<CIM_TCPProtocolEndpoint single instance>
1507  &smOpDeleteInstance ( $instance.GetObjectPath() );
1508  &smEnd;
```

1509 **6.12.4   Set**

1510 This section describes how to implement the set verb when it is applied to an instance of
1511 CIM_TCPProtocolEndpoint. Implementations may support the use of the set verb with
1512 CIM_TCPProtocolEndpoint.

1513 The set verb is used to modify descriptive properties of the CIM_TCPProtocolEndpoint instance.

1514 **6.12.4.1   General Usage of Set for a Single Property**

1515 This command form corresponds to the general usage of the set verb to modify a single property of a
1516 target instance. This is the most common case.

1517 The requirement for supporting modification of a property using this command form shall be equivalent to
1518 the requirement for supporting modification of the property using the ModifyInstance operation as defined
1519 in the *Telnet Service Profile*.

1520 **6.12.4.1.1   Command Form**

```
1521  set <CIM_TCPProtocolEndpoint single instance> <propertyname>=<propertyvalue>
```

1522 **6.12.4.1.2   CIM Requirements**

1523 See CIM_TCPProtocolEndpoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1524 modifiable properties.

1525 **6.12.4.1.3   Behavior Requirements**

```
1526  $instance=<CIM_TCPProtocolEndpoint single instance>
1527  #propertyNames[] = {<propertyname>};
1528  #propertyValues[] = {<propertyvalue>};
1529  &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
1530  &smEnd;
```

1531 **6.12.4.2   General Usage of Set for Multiple Properties**

1532 This command form corresponds to the general usage of the set verb to modify multiple properties of a
1533 target instance where there is not an explicit relationship between the properties. This is the most
1534 common case.

1535 The requirement for supporting modification of a property using this command form shall be equivalent to
1536 the requirement for supporting modification of the property using the ModifyInstance operation as defined
1537 in the *Telnet Service Profile*.

1538 **6.12.4.2.1   Command Form**

```
1539  set <CIM_TCPProtocolEndpoint single instance> <propertyname1>=<propertyvalue1>
1540     <propertynamen>=<propertyvaluen>
```

1541 **6.12.4.2.2 CIM Requirements**

1542 See CIM_TCPProtocolEndpoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1543 mandatory properties.

1544 **6.12.4.2.3 Behavior Requirements**

```
1545   $instance=<CIM_TCPProtocolEndpoint single instance>
1546   #propertyNames[] = {<propertyname>};
1547   for #i < n
1548       {
1549       #propertyNames[#i] = <propertname#i>
1550       #propertyValues[#i] = <propertyvalue#i>
1551       }
1552   &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
1553   &smEnd;
```

1554 **6.12.5 Show**

1555 This section describes how to implement the show verb when applied to an instance of
1556 CIM_TCPProtocolEndpoint. Implementations shall support the use of the show verb with
1557 CIM_TCPProtocolEndpoint.

1558 The show verb is used to display information about a CIM_TCPProtocolEndpoint instance.

1559 **6.12.5.1 Show a Single Instance**

1560 This command form is for the show verb applied to a single instance of CIM_TCPProtocolEndpoint.

1561 **6.12.5.1.1 Command Form**

```
1562   show <CIM_TCPProtocolEndpoint single instance>
```

1563 **6.12.5.1.2 CIM Requirements**

1564 See CIM_TCPProtocolEndpoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1565 mandatory properties.

1566 **6.12.5.1.3 Behavior Requirements**

1567 **6.12.5.1.3.1 Preconditions**

1568 #all is true if the "-all" option was specified with the command; otherwise, #all is false.

1569 **6.12.5.1.3.2 Pseudo Code**

```
1570   $instance=<CIM_TCPProtocolEndpoint single instance>
1571   &lShowTCPEndpoint ( $instance, #all );
1572   &smEnd;
```

1573 **6.12.5.2 Show Multiple Instances Scoped by a System**

1574 This command form is for the show verb applied to multiple instances of CIM_TCPProtocolEndpoint. This
1575 command form corresponds to UFsT-based selection within a scoping system.

1576 **6.12.5.2.1   Command Form**

1577 `show <CIM_TCPProtocolEndpoint multiple instances>`

1578 **6.12.5.2.2   CIM Requirements**

1579 See CIM_TCPProtocolEndpoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1580 mandatory properties.

1581 **6.12.5.2.3   Behavior Requirements**

1582 **6.12.5.2.3.1   Preconditions**

1583 `$containerInstance` contains the instance of CIM_ComputerSystem for which scoped endpoints
1584 (CIM_TCPProtocolEndpoint instances) are displayed. The *Telnet Service Profile* requires that the
1585 CIM_TCPProtocolEndpoint instance be associated with its scoping system via an instance of the
1586 CIM_HostedAccessPoint association.

1587 `#all` is true if the "-all" option was specified with the command; otherwise, `#all` is false.

1588 **6.12.5.2.3.2   Pseudo Code**

```
1589 #propertylist[] = NULL;
1590 //this property list will match the property list in lShowTCPEndpoint()
1591 if (false == #all)
1592     {
1593     #propertylist[] = { //all mandatory non-key properties };
1594     }
1595 &smShowInstances ( "CIM_TCPProtocolEndpoint", "CIM_HostedAccessPoint",
1596     $containerInstance.getObjectPath(), #propertylist[] );
1597 &smEnd;
```

1598 **6.12.5.3   Show Multiple Instances Scoped by a ProtocolService**

1599 This command form is for the `show` verb applied to multiple instances of CIM_TCPProtocolEndpoint. This
1600 command form corresponds to UFsT-based selection within a scoping ProtocolService instance.

1601 **6.12.5.3.1   Command Form**

1602 `show <CIM_TCPProtocolEndpoint multiple instances>`

1603 **6.12.5.3.2   CIM Requirements**

1604 See CIM_TCPProtocolEndpoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1605 mandatory properties.

1606 **6.12.5.3.3   Behavior Requirements**

1607 **6.12.5.3.3.1   Preconditions**

1608 `$containerInstance` contains the instance of CIM_ProtocolService for which associated endpoints
1609 (CIM_TCPProtocolEndpoint instances) are displayed. The *Telnet Service Profile* requires that the
1610 CIM_TCPProtocolEndpoint instance be associated with an instance of CIM_ProtocolService via an
1611 instance of CIM_ServiceAccessBySAP.

1612 `#all` is true if the "-all" option was specified with the command; otherwise, `#all` is false.

1613   **6.12.5.3.3.2   Pseudo Code**

```
1614    #propertylist[] = NULL;
1615    //this property list will match the property list in lShowTCPEndpoint()
1616    if (false == #all)
1617        {
1618        #propertylist[] = { //all mandatory non-key properties };
1619        }
1620    &smShowInstances ( "CIM_TCPProtocolEndpoint", "CIM_ServiceAccessBySAP",
1621        $containerInstance.getObjectPath(), #propertylist[] );
1622    &smEnd;
```

1623   **6.12.5.4   Show Multiple Instances Scoped by a ProtocolEndpoint**

1624   This command form is for the show verb applied to multiple instances of CIM_TCPProtocolEndpoint. This
1625   command form corresponds to UFsT-based selection within a scoping CIM_ProtocolEndpoint instance
1626   with which the CIM_TCPProtocolEndpoint instances are associated via instances of CIM_BindsTo.

1627   **6.12.5.4.1   Command Form**

1628   **show <CIM_TCPProtocolEndpoint *multiple instances*>**

1629   **6.12.5.4.2   CIM Requirements**

1630   See CIM_TCPProtocolEndpoint in the "CIM Elements" section of the *Telnet Service Profile* for the list of
1631   mandatory properties.

1632   **6.12.5.4.3   Behavior Requirements**

1633   **6.12.5.4.3.1   Preconditions**

1634 $containerInstance contains the instance of CIM_ProtocolEndpoint for which associated endpoints
1635   (CIM_TCPProtocolEndpoint instances) are displayed. The *Telnet Service Profile* indicates that the
1636   CIM_TCPProtocolEndpoint instance can be associated with an instance of CIM_ProtocolEndpoint via an
1637   instance of CIM_BindsTo.

1638   #all is true if the "-all" option was specified with the command; otherwise, #all is false.

1639   **6.12.5.4.3.2   Pseudo Code**

```
1640    #propertylist[] = NULL;
1641    //this property list will match the property list in lShowTCPEndpoint()
1642    if (false == #all)
1643        {
1644        #propertylist[] = { //all mandatory non-key properties };
1645        }
1646    &smShowInstances ( "CIM_TCPProtocolEndpoint", "CIM_BindsTo",
1647        $containerInstance.getObjectPath(), #propertylist[] );
1648    &smEnd;
```

1649

1650
ANNEX A
1651
(informative)
1652

1653

1654
Change Log

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

1655