



1  
2  
3  
4

**Document Number: DSP0810**

**Date: 2009-07-14**

**Version: 1.0.0**

5 **Record Log Profile SM CLP Mapping**  
6 **Specification**

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

10

11 Copyright notice

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

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

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27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party  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

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

33

## CONTENTS

34	Foreword .....	5
35	Introduction .....	6
36	1 Scope .....	7
37	2 Normative References.....	7
38	2.1 Approved References .....	7
39	2.2 Other References.....	7
40	3 Terms and Definitions.....	7
41	4 Symbols and Abbreviated Terms.....	8
42	5 Recipes.....	9
43	6 Mappings.....	9
44	6.1 CIM_ElementCapabilities .....	9
45	6.2 CIM_EnabledLogicalElementCapabilities.....	12
46	6.3 CIM_LogManagesRecord.....	14
47	6.4 CIM_LogEntry .....	17
48	6.5 CIM_RecordLog.....	22
49	6.6 CIM_UseOfLog .....	26
50	ANNEX A (informative) Change Log.....	30

51

## 52 Tables

53	Table 1 – Command Verb Requirements for CIM_ElementCapabilities .....	9
54	Table 2 – Command Verb Requirements for CIM_EnabledLogicalElementCapabilities.....	12
55	Table 3 – Command Verb Requirements for CIM_LogManagesRecord.....	14
56	Table 4 – Command Verb Requirements for CIM_LogEntry .....	17
57	Table 5 – Command Verb Requirements for CIM_RecordLog.....	22
58	Table 6 – Command Verb Requirements for CIM_UseOfLog .....	27

59



61

## Foreword

62 The *Record Log Profile SM CLP Mapping Specification* (DSP0810) was prepared by the Server  
63 Management Working Group.

### 64 **Conventions**

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

### 67 **Acknowledgements**

68 The authors wish to acknowledge the following participants from the DMTF Server Management Working  
69 Group:

- 70 • Khachatur Papanyan – Dell Inc.
- 71 • Jon Hass – Dell, Inc.
- 72 • Jeff Hilland – HP
- 73 • Christina Shaw – HP
- 74 • Aaron Merkin – IBM
- 75 • Jeff Lynch – IBM
- 76 • Perry Vincent – Intel
- 77 • John Leung – Intel

78

79

## Introduction

80 This document defines the SM CLP mapping for CIM elements described in the [Record Log Profile](#). The  
81 information in this specification, combined with the [SM CLP-to-CIM Common Mapping Specification 1.0](#),  
82 is intended to be sufficient to implement SM CLP commands relevant to the classes, properties, and  
83 methods described in the [Record Log Profile](#) using CIM operations.

84 The target audience for this specification is implementers of the SM CLP support for the [Record Log](#)  
85 [Profile](#).

86

# Record Log Profile SM CLP Mapping Specification

## 1 Scope

88 This specification contains the requirements for an implementation of the SM CLP to provide access to,  
89 and implement the behaviors of, the [Record Log Profile](#).

## 2 Normative References

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

### 2.1 Approved References

95 DMTF DSP0216, *SM CLP-to-CIM Common Mapping Specification 1.0*,  
96 [http://www.dmtf.org/standards/published\\_documents/DSP0216\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0216_1.0.pdf)

97 DMTF DSP1010, *Record Log Profile 1.0*,  
98 [http://www.dmtf.org/standards/published\\_documents/DSP1010\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1010_1.0.pdf)

99 SNIA, *Storage Management Initiative Specification (SMI-S) 1.1.0*,  
100 [http://www.snia.org/tech\\_activities/standards/curr\\_standards/smi](http://www.snia.org/tech_activities/standards/curr_standards/smi)

### 2.2 Other References

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

## 3 Terms and Definitions

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

### 3.1

#### **can**

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

### 3.2

#### **cannot**

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

### 3.3

#### **conditional**

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

### 3.4

#### **mandatory**

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

- 120 **3.5**  
121 **may**  
122 indicates a course of action permissible within the limits of the document
- 123 **3.6**  
124 **need not**  
125 indicates a course of action permissible within the limits of the document
- 126 **3.7**  
127 **optional**  
128 indicates a course of action permissible within the limits of the document
- 129 **3.8**  
130 **shall**  
131 indicates requirements to be followed strictly in order to conform to the document and from which no  
132 deviation is permitted
- 133 **3.9**  
134 **shall not**  
135 indicates requirements to be followed strictly in order to conform to the document and from which no  
136 deviation is permitted
- 137 **3.10**  
138 **should**  
139 indicates that among several possibilities, one is recommended as particularly suitable, without  
140 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 141 **3.11**  
142 **should not**  
143 indicates that a certain possibility or course of action is deprecated but not prohibited

## 144 **4 Symbols and Abbreviated Terms**

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

- 146 **4.1**  
147 **CIM**  
148 Common Information Model
- 149 **4.2**  
150 **CLP**  
151 Command Line Protocol
- 152 **4.3**  
153 **DMTF**  
154 Distributed Management Task Force
- 155 **4.4**  
156 **IETF**  
157 Internet Engineering Task Force



158 **4.5**  
 159 **SM**  
 160 Server Management

161 **4.6**  
 162 **SMI-S**  
 163 Storage Management Initiative Specification

164 **4.7**  
 165 **SNIA**  
 166 Storage Networking Industry Association

## 167 **5 Recipes**

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

- 170 • smResetRSC
- 171 • smShowInstance
- 172 • smShowInstances
- 173 • smShowAssociationInstance
- 174 • smShowAssociationInstances
- 175 • smStartRSC
- 176 • smStopRSC

## 177 **6 Mappings**

178 The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in  
 179 the [Record Log Profile](#). Requirements specified here related to support for a CLP verb for a particular  
 180 class are solely within the context of this profile.

### 181 **6.1 CIM\_ElementCapabilities**

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

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

188 **Table 1 – 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	

Command Verb	Requirement	Comments
Set	Not supported	
Show	Shall	See 6.1.2
Start	Not supported	
Stop	Not supported	

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

### 191 6.1.1 Ordering of Results

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

- 194 • Results for CIM\_ElementCapabilities are unordered; therefore, no algorithm is defined.

### 195 6.1.2 Show

196 This section describes how to implement the `show` verb when applied to an instance of  
197 CIM\_ElementCapabilities. Implementations shall support the use of the `show` verb with  
198 CIM\_ElementCapabilities.

#### 199 6.1.2.1 Show Command Form for Multiple Instances Target – 200 CIM\_EnabledLogicalElementCapabilities Reference

201 This command form is used to show many instances of CIM\_ElementCapabilities. This command form  
202 corresponds to a `show` command issued against instances of CIM\_ElementCapabilities where only one  
203 reference is specified and the reference is to the scoping instance of  
204 CIM\_EnabledLogicalElementCapabilities.

##### 205 6.1.2.1.1 Command Form

```
206 show <CIM_ElementCapabilities multiple instances>
```

##### 207 6.1.2.1.2 CIM Requirements

208 See CIM\_ElementCapabilities in the “CIM Elements” section of the [Record Log Profile](#) for the list of  
209 mandatory properties.

##### 210 6.1.2.1.3 Behavior Requirements

###### 211 6.1.2.1.3.1 Preconditions

212 In this section `$instance` represents the instance of CIM\_EnabledLogicalElementCapabilities which is  
213 referenced by CIM\_ElementCapabilities.

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

###### 215 6.1.2.1.3.2 Pseudo Code

```
216 #propertylist[] = NULL;
217 if ( false == #all)
218 {
219     #propertylist[] = <array of mandatory non-key property names (see CIM
220     Requirements)>;
221 }
```

```

222 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getInstancePath(),
223     #propertylist[] );
224 &smEnd;

```

### 225 6.1.2.2 Show Command Form for a Single Instance – CIM\_RecordLog Reference

226 This command form is used to show a single instance of CIM\_ElementCapabilities. This command form  
 227 corresponds to a `show` command issued against a single instance of CIM\_ElementCapabilities where  
 228 only one reference is specified and the reference is to the instance of CIM\_RecordLog.

#### 229 6.1.2.2.1 Command Form

```

230 show <CIM_ElementCapabilities single instance>

```

#### 231 6.1.2.2.2 CIM Requirements

232 See CIM\_ElementCapabilities in the “CIM Elements” section of the [Record Log Profile](#) for the list of  
 233 mandatory properties.

#### 234 6.1.2.2.3 Behavior Requirements

##### 235 6.1.2.2.3.1 Preconditions

236 In this section `$instance` represents the instance of CIM\_RecordLog which is referenced by  
 237 CIM\_ElementCapabilities.

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

##### 239 6.1.2.2.3.2 Pseudo Code

```

240 #propertylist[] = NULL;
241 if ( false == #all)
242 {
243     #propertylist[] = <array of mandatory non-key property names (see CIM
244         Requirements)>;
245 }
246 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getInstancePath(),
247     #propertylist[] );
248 &smEnd;

```

### 249 6.1.2.3 Show Command Form for a Single Instance Target – Both References

250 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
 251 a `show` command issued against CIM\_ElementCapabilities where both references are specified and  
 252 therefore the desired instance is unambiguously identified.

#### 253 6.1.2.3.1 Command Form

```

254 show <CIM_ElementCapabilities single instance>

```

#### 255 6.1.2.3.2 CIM Requirements

256 See CIM\_ElementCapabilities in the “CIM Elements” section of the [Record Log Profile](#) for the list of  
 257 mandatory properties.

258 **6.1.2.3.3 Behavior Requirements**259 **6.1.2.3.3.1 Preconditions**

260 In this section `$instanceA` represents the referenced instance of `CIM_RecordLog` through the  
 261 `CIM_ElementCapabilities` association. `$instanceB` represents the instance of  
 262 `CIM_EnabledLogicalElementCapabilities` which is referenced by `CIM_ElementCapabilities`.

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

264 **6.1.2.3.3.2 Pseudo Code**

```

265 #propertylist[] = NULL;
266 if ( false == #all)
267 {
268     #propertylist[] = <array of mandatory non-key property names (see CIM
269         Requirements)>;
270 }
271 &smShowAssociationInstance ( "CIM_ElementCapabilities", $instanceA.getInstancePath(),
272     $instanceB.getInstancePath(), #propertylist[] );
273 &smEnd;
  
```

274 **6.2 CIM\_EnabledLogicalElementCapabilities**

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

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

281 **Table 2 – Command Verb Requirements for CIM\_EnabledLogicalElementCapabilities**

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	

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

## 284 6.2.1 Ordering of Results

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

- 287 • Results for CIM\_EnabledLogicalElementCapabilities are unordered; therefore, no algorithm is  
288 defined.

## 289 6.2.2 Show

290 This section describes how to implement the `show` verb when applied to an instance of  
291 CIM\_EnabledLogicalElementCapabilities. Implementations shall support the use of the `show` verb with  
292 CIM\_EnabledLogicalElementCapabilities.

### 293 6.2.2.1 Show Command Form for Multiple Instances Target

294 This command form is used to show many instances of CIM\_EnabledLogicalElementCapabilities.

#### 295 6.2.2.1.1 Command Form

```
296 show <CIM_EnabledLogicalElementCapabilities multiple instances>
```

#### 297 6.2.2.1.2 CIM Requirements

298 See CIM\_EnabledLogicalElementCapabilities in the “CIM Elements” section of the [Record Log Profile](#) for  
299 the list of mandatory properties.

#### 300 6.2.2.1.3 Behavior Requirements

##### 301 6.2.2.1.3.1 Preconditions

302 In this section `$containerInstance` represents the instance of CIM\_ConcreteCollection with  
303 ElementName property that contains “Capabilities” and is associated to the targeted instances of  
304 CIM\_EnabledLogicalElementCapabilities through the CIM\_MemberOfCollection association.

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

##### 306 6.2.2.1.3.2 Pseudo Code

```
307 #propertylist[] = NULL;
308 if ( false == #all)
309 {
310     #propertylist[] = <array of mandatory non-key property names (see CIM
311         Requirements)>;
312 }
313 &smShowInstances ( "CIM_EnabledLogicalElementCapabilities", "CIM_MemberOfCollection",
314     $containerInstance.getInstancePath(), #propertylist[] );
315 &smEnd;
```

### 316 6.2.2.2 Show Command Form for a Single Instance Target

317 This command form is used to show a single instance of CIM\_EnabledLogicalElementCapabilities.

#### 318 6.2.2.2.1 Command Form

```
319 show <CIM_EnabledLogicalElementCapabilities single instance>
```

320 **6.2.2.2.2 CIM Requirements**

321 See CIM\_EnabledLogicalElementCapabilities in the “CIM Elements” section of the [Record Log Profile](#) for  
322 the list of mandatory properties.

323 **6.2.2.2.3 Behavior Requirements**324 **6.2.2.2.3.1 Preconditions**

325 In this section \$instance represents the targeted instance of CIM\_EnabledLogicalElementCapabilities.

```
326 $instance=<CIM_EnabledLogicalElementCapabilities single instance>;
```

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

328 **6.2.2.2.3.2 Pseudo Code**

```
329 #propertylist[] = NULL;
330 if ( false == #all)
331 {
332     #propertylist[] = <array of mandatory non-key property names (see CIM
333     Requirements)>;
334 }
335 &smShowInstance ( $instance.getInstancePath(), #propertylist[] );
336 &smEnd;
```

337 **6.3 CIM\_LogManagesRecord**

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

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

344 **Table 3 – Command Verb Requirements for CIM\_LogManagesRecord**

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 6.3.2.
Start	Not supported	
Stop	Not supported	

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

### 347 6.3.1 Ordering of Results

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

- 350 • Results for CIM\_LogManagesRecord are unordered; therefore, no algorithm is defined.

### 351 6.3.2 Show

352 This section describes how to implement the `show` verb when applied to an instance of  
353 CIM\_LogManagesRecord. Implementations shall support the use of the `show` verb with  
354 CIM\_LogManagesRecord.

#### 355 6.3.2.1 Show Command Form for Multiple Instances Target – CIM\_RecordLog Reference

356 This command form is used to show many instances of CIM\_LogManagesRecord. This command form  
357 corresponds to a `show` command issued against instances of CIM\_LogManagesRecord where only one  
358 reference is specified and the reference is to the scoping instance of CIM\_RecordLog

##### 359 6.3.2.1.1 Command Form

```
360 show <CIM_LogManagesRecord multiple instances>
```

##### 361 6.3.2.1.2 CIM Requirements

362 See CIM\_LogManagesRecord in the “CIM Elements” section of the [Record Log Profile](#) for the list of  
363 mandatory properties.

##### 364 6.3.2.1.3 Behavior Requirements

###### 365 6.3.2.1.3.1 Preconditions

366 In this section `$instance` represents the instance of CIM\_RecordLog which is referenced by  
367 CIM\_LogManagesRecord.

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

###### 369 6.3.2.1.3.2 Pseudo Code

```
370 #propertylist[] = NULL;
371 if ( false == #all)
372 {
373     #propertylist[] = <array of mandatory non-key property names (see CIM
374     Requirements)>;
375 }
376 &smShowAssociationInstances ( "CIM_LogManagesRecord", $instance.getInstancePath(),
377     #propertylist[] );
378 &smEnd;
```

#### 379 6.3.2.2 Show Command Form for a Single Instance – CIM\_LogEntry Reference

380 This command form is used to show a single instance of CIM\_LogManagesRecord. This command form  
381 corresponds to a `show` command issued against a single instance of CIM\_LogManagesRecord where  
382 only one reference is specified and the reference is to the instance of CIM\_LogEntry.

##### 383 6.3.2.2.1 Command Form

```
384 show <CIM_LogManagesRecord single instance>
```

### 385 6.3.2.2.2 CIM Requirements

386 See CIM\_LogManagesRecord in the “CIM Elements” section of the [Record Log Profile](#) for the list of  
387 mandatory properties.

### 388 6.3.2.2.3 Behavior Requirements

#### 389 6.3.2.2.3.1 Preconditions

390 In this section `$instance` represents the instance of CIM\_LogEntry which is referenced by  
391 CIM\_LogManagesRecord.

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

#### 393 6.3.2.2.3.2 Pseudo Code

```
394 #propertylist[] = NULL;
395 if ( false == #all)
396 {
397     #propertylist[] = <array of mandatory non-key property names (see CIM
398     Requirements)>;
399 }
400 &smShowAssociationInstances ( "CIM_LogManagesRecord", $instance.getInstancePath(),
401     #propertylist[] );
402 &smEnd;
```

### 403 6.3.2.3 Show Command Form for a Single Instance Target – Both References

404 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
405 a `show` command issued against CIM\_LogManagesRecord where both references are specified and  
406 therefore the desired instance is unambiguously identified.

#### 407 6.3.2.3.1 Command Form

```
408 show <CIM_LogManagesRecord single instance>
```

#### 409 6.3.2.3.2 CIM Requirements

410 See CIM\_LogManagesRecord in the “CIM Elements” section of the [Record Log Profile](#) for the list of  
411 mandatory properties.

#### 412 6.3.2.3.3 Behavior Requirements

##### 413 6.3.2.3.3.1 Preconditions

414 In this section `$instanceA` represents the referenced instance of CIM\_RecordLog through  
415 CIM\_LogManagesRecord association. `$instanceB` represents the instance of CIM\_LogEntry which is  
416 referenced by CIM\_LogManagesRecord.

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



418 **6.3.2.3.3.2 Pseudo Code**

```

419 #propertylist[] = NULL;
420 if ( false == #all)
421 {
422     #propertylist[] = <array of mandatory non-key property names (see CIM
423     Requirements)>;
424 }
425 &smShowAssociationInstance ( "CIM_LogManagesRecord", $instanceA.getInstancePath(),
426     $instanceB.getInstancePath(), #propertylist[] );
427 &smEnd;
    
```

428 **6.4 CIM\_LogEntry**

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

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

435 **Table 4 – Command Verb Requirements for CIM\_LogEntry**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	May	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.4.3.
Start	Not supported	
Stop	Not supported	

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

437 **6.4.1 Ordering of Results**

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

- 440 • Results for `CIM_LogEntry` instances are based on the `TimeStamp` property of the  
 441 `CIM_LogEntry` class. The natural order is descending: LIFO (Last In, First Out), from the newest  
 442 log entry to the oldest. In the rare case when `TimeStamp` values for some `CIM_LogEntry`  
 443 instances are equal, it is left up to the implementation to sort those instances.

444 **6.4.2 Delete**

445 This section describes how to implement the `delete` verb when applied to an instance of `CIM_LogEntry`.  
 446 Implementations may support the use of the `delete` verb with `CIM_LogEntry`. Deleting all the instances

447 of CIM\_LogEntry aggregated within a CIM\_RecordLog instance corresponds to clearing the log. Deleting  
 448 a single instance of CIM\_LogEntry corresponds to removing an entry from the log. Implementation may  
 449 support one or both of the below command forms.

#### 450 6.4.2.1 Delete Multiple Instances – All Instance of Log Entries

451 Used to clear all the log entries as all CIM\_LogEntry instances contained within the CIM\_RecordLog are  
 452 the target.

##### 453 6.4.2.1.1 Command Form

```
454 delete <CIM_LogEntry multiple instances>
```

##### 455 6.4.2.1.2 CIM Requirements

```
456 uint32 CIM_RecordLog.ClearLog()
```

##### 457 6.4.2.1.3 Behavior Requirements

###### 458 6.4.2.1.3.1 Preconditions

459 In this section \$containerInstance is the instance of CIM\_RecordLog, which represents the container  
 460 log for the entries to be deleted, and is associated to the targeted instances of CIM\_LogEntry through the  
 461 CIM\_LogManagesRecord association.

###### 462 6.4.2.1.3.2 Pseudo Code

```
463 #Error = &smOpEnumerateInstances ( "CIM_LogEntry", $deleteInstances[] );
464 if ( 0 != #Error.code)
465 {
466     //includes &smEnd;
467     &smProcessOpError (#Error);
468 }
469 %InArguments[] = {};
470 %OutArguments[] = {};
471 #Error = InvokeMethod ($containerInstance.getInstancePath(),
472     "ClearLog",
473     %InArguments[],
474     %OutArguments[],
475     #returnStatus);
476 if ( 0 != #Error.code)
477 {
478     //method invocation failed
479     if ( (null != #Error.$error) && (null != #Error.$error[0]) )
480     {
481         //if the method invocation contains an embedded error
482         //use it for the Error for the overall job
483         &smAddError($job, #Error.$error[0]);
484         &smMakeCommandStatus($job);
485         &smEnd;
486     }
487     else if ( 17 == #Error.code )
488     {
489         //17 - CIM_ERR_METHOD_NOT_FOUND
490         // The specified extrinsic method does not exist.
491         $OperationError = smNewInstance("CIM_Error");
```

```
492     // CIM_ERR_METHOD_NOT_FOUND
493     $OperationError.CIMStatusCode = 17;
494     //Software Error
495     $OperationError.ErrorType = 10;
496     //Unknown
497     $OperationError.PerceivedSeverity = 0;
498     $OperationError.OwningEntity = DMTF:SMCLP;
499     $OperationError.MessageID = 0x00000001;
500     $OperationError.Message = "Operation is not supported."
501     &smAddError($job, $OperationError);
502     &smMakeCommandStatus($job);
503     &smEnd;
504 }
505 else
506 {
507     //operation failed, but no detailed error instance, need to make one up
508     //make an Error instance and associate with job for Operation
509     $OperationError = smNewInstance("CIM_Error");
510     //CIM_ERR_FAILED
511     $OperationError.CIMStatusCode = 1;
512     //Software Error
513     $OperationError.ErrorType = 4;
514     //Unknown
515     $OperationError.PerceivedSeverity = 0;
516     $OperationError.OwningEntity = DMTF:SMCLP;
517     $OperationError.MessageID = 0x00000009;
518     $OperationError.Message = "An internal software error has occurred.";
519     &smAddError($job, $OperationError);
520     &smMakeCommandStatus($job);
521     &smEnd;
522 }
523 }
524 //if CIM op failed
525 else if (0 == #returnStatus)
526 {
527     //completed successfully
528     &smCommandCompleted($job);
529     &smEnd;
530 }
531 else if (1 == #returnStatus)
532 {
533     //unsupported
534     $OperationError = smNewInstance("CIM_Error");
535     //CIM_ERR_NOT_SUPPORTED
536     $OperationError.CIMStatusCode = 7;
537     //Other
538     $OperationError.ErrorType = 1;
539     //Low
540     $OperationError.PerceivedSeverity = 2;
541     $OperationError.OwningEntity = DMTF:SMCLP;
542     $OperationError.MessageID = 0x00000001;
543     $OperationError.Message = "Operation is not supported.";
544     &smAddError($job, $OperationError);
```

```

545     &smMakeCommandStatus($job);
546     &smEnd;
547 }
548 else
549 {
550     // generic failure
551     $OperationError = smNewInstance("CIM_Error");
552     //CIM_ERR_FAILED
553     $OperationError.CIMStatusCode = 1;
554     //Other
555     $OperationError.ErrorType = 1;
556     //Low
557     $OperationError.PerceivedSeverity = 2;
558     $OperationError.OwningEntity = DMTF:SMCLP;
559     $OperationError.MessageID = 0x00000002;
560     $OperationError.Message = "Failed. No further information is available.";
561     &smAddError($job, $OperationError);
562     &smMakeCommandStatus($job);
563     &smEnd;
564 }
565 for #i in $deleteInstances[]
566 {
567     &smDisplayInstance ( $deleteInstances[#i] );
568 }
569 &smEnd;

```

#### 570 6.4.2.2 Delete a Single Instance

571 This command is used to delete a single entry within the log.

##### 572 6.4.2.2.1 Command Form

```
573 delete <CIM_LogEntry single instance>
```

##### 574 6.4.2.2.2 CIM Requirements

575 See CIM\_LogEntry in the "CIM Elements" section of the [Record Log Profile](#) for the list of mandatory  
576 properties.

##### 577 6.4.2.2.3 Behavior Requirements

###### 578 6.4.2.2.3.1 Preconditions

```
579 $instanceLogRecord=<CIM_LogEntry single instance>;
```

###### 580 6.4.2.2.3.2 Pseudo Code

```

581 #Error = &smOpDeleteInstance ( $instanceLogRecord.getInstancePath() );
582 //deletes association classes referencing the instance as well
583 if (0 != Error.code)
584 {
585     &smProcessOpError (#Error);
586     //includes end;
587 }
588 &smDisplayInstance ( $instanceLogRecord );
589 &smEnd;

```

### 590 **6.4.3 Show**

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

#### 593 **6.4.3.1 Show Command Form for Multiple Instances Target**

594 This command form is used to show many instances of `CIM_LogEntry`.

##### 595 **6.4.3.1.1 Command Form**

```
596 show <CIM_LogEntry multiple instances>
```

##### 597 **6.4.3.1.2 CIM Requirements**

598 See `CIM_LogEntry` in the “CIM Elements” section of the [Record Log Profile](#) for the list of mandatory  
 599 properties.

##### 600 **6.4.3.1.3 Behavior Requirements**

###### 601 **6.4.3.1.3.1 Preconditions**

602 In this section `$containerInstance` represents the instance of `CIM_RecordLog` which represents the  
 603 log and is associated to the targeted instances of `CIM_LogEntry` through the `CIM_LogManagesRecord`  
 604 association.

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

###### 606 **6.4.3.1.3.2 Pseudo Code**

```
607 #propertylist[] = NULL;
608 if ( false == #all)
609 {
610     #propertylist[] = <array of mandatory non-key property names (see CIM
611         Requirements)>;
612 }
613 &smShowInstances ( "CIM_LogEntry", "CIM_LogManagesRecord",
614     $containerInstance.getInstancePath(), #propertylist[] );
615 &smEnd;
```

#### 616 **6.4.3.2 Show Command Form for a Single Instance Target**

617 This command form is used to show a single instance of `CIM_LogEntry`.

##### 618 **6.4.3.2.1 Command Form**

```
619 show <CIM_LogEntry single instance>
```

##### 620 **6.4.3.2.2 CIM Requirements**

621 See `CIM_LogEntry` in the “CIM Elements” section of the [Record Log Profile](#) for the list of mandatory  
 622 properties.

##### 623 **6.4.3.2.3 Behavior Requirements**

###### 624 **6.4.3.2.3.1 Preconditions**

625 In this section `$instance` represents the targeted instance of `CIM_LogEntry`.

626 `$instance=<CIM_LogEntry single instance>;`

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

#### 628 **6.4.3.2.3.2 Pseudo Code**

```
629 #propertylist[] = NULL;
630 if ( false == #all)
631 {
632     #propertylist[] = <array of mandatory non-key property names (see CIM
633     Requirements)>;
634 }
635 &smShowInstance ( $instance.getInstancePath(), #propertylist[] );
636 &smEnd;
```

### 637 **6.5 CIM\_RecordLog**

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

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

644 **Table 5 – Command Verb Requirements for CIM\_RecordLog**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	May	See 6.5.2.
Set	May	See 6.5.3.
Show	Shall	See 6.5.4.
Start	May	See 6.5.5.
Stop	May	See 6.5.6.

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

#### 646 **6.5.1 Ordering of Results**

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

- 649 • Results for `CIM_RecordLog` are unordered; therefore, no algorithm is defined.

#### 650 **6.5.2 Reset**

651 This section describes how to implement the `reset` verb when applied to an instance of  
 652 `CIM_RecordLog`. Implementations may support the use of the `reset` verb with `CIM_RecordLog`.

### 653 6.5.2.1 Command Form

```
654 reset <CIM_RecordLog single instance>
```

### 655 6.5.2.2 CIM Requirements

```
656 uint16 EnabledState;
657 uint16 RequestedState;
658 uint32 CIM_RecordLog.RequestStateChange (
659     [IN] uint16 RequestedState,
660     [OUT] REF CIM_ConcreteJob Job,
661     [IN] datetime TimeoutPeriod );
```

### 662 6.5.2.3 Behavior Requirements

#### 663 6.5.2.3.1.1 Preconditions

664 In this section `$instance` represents the targeted instance of `CIM_RecordLog`.

```
665 $instance=<CIM_RecordLog single instance>;
```

#### 666 6.5.2.3.2 Pseudo Code

```
667 &smResetRSC ( $instance.getInstancePath() );
668 &smEnd;
```

### 669 6.5.3 Set

670 This section describes how to implement the `set` verb when it is applied to an instance of  
671 `CIM_RecordLog`. Implementations may support the use of the `set` verb with `CIM_RecordLog`.

672 The `set` verb is used to modify descriptive properties of the `CIM_RecordLog` instance.

#### 673 6.5.3.1 General Usage of Set for a Single Property

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

676 The requirement for supporting modification of a property using this command form shall be equivalent to  
677 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined  
678 in the [Record Log Profile](#).

#### 679 6.5.3.1.1 Command Form

```
680 set <CIM_RecordLog single instance> <propertyname>=<propertyvalue>
```

#### 681 6.5.3.1.2 CIM Requirements

682 See `CIM_RecordLog` in the “CIM Elements” section of the [Record Log Profile](#) for the list of mandatory  
683 properties.

#### 684 6.5.3.1.3 Behavior Requirements

```
685 $instance=<CIM_RecordLog single instance>
686 #propertyName[] = {<propertyname>};
687 #propertyValues[] = {<propertyvalue>};
688 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
689 &smEnd;
```

### 690 6.5.3.2 General Usage of Set for Multiple Properties

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

694 The requirement for supporting modification of a property using this command form shall be equivalent to  
695 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined  
696 in the [Record Log Profile](#).

#### 697 6.5.3.2.1 Command Form

```
698 set <CIM_RecordLog single instance> <propertyname1>=<propertyvalue1>  
699 <propertynamen>=<propertyvaluen>
```

#### 700 6.5.3.2.2 CIM Requirements

701 See `CIM_RecordLog` in the “CIM Elements” section of the [Record Log Profile](#) for the list of mandatory  
702 properties.

#### 703 6.5.3.2.3 Behavior Requirements

```
704 $instance=<CIM_RecordLog single instance>  
705 #propertyName[] = {<propertyname>};  
706 for #i < n  
707 {  
708     #propertyName[#i] = <propertyname#i>  
709     #propertyValue[#i] = <propertyvalue#i>  
710 }  
711 &smSetInstance ( $instance, #propertyName[], #propertyValue[] );  
712 &smEnd;
```

### 713 6.5.4 Show

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

#### 716 6.5.4.1 Show Command Form for Multiple Instances Target

717 This command form is used to show many instances of `CIM_RecordLog`.

##### 718 6.5.4.1.1 Command Form

```
719 show <CIM_RecordLog multiple instances>
```

##### 720 6.5.4.1.2 CIM Requirements

721 See `CIM_RecordLog` in the “CIM Elements” section of the [Record Log Profile](#) for the list of mandatory  
722 properties.

##### 723 6.5.4.1.3 Behavior Requirements

###### 724 6.5.4.1.3.1 Preconditions

725 In this section `$containerInstance` represents the instance of `CIM_ConcreteCollection` which  
726 represents the collection of logs and is associated to the targeted instances of `CIM_RecordLog` through  
727 the `CIM_MemberOfCollection` association.

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



### 729 6.5.4.1.3.2 Pseudo Code

```

730 #propertylist[] = NULL;
731 if ( false == #all)
732 {
733     #propertylist[] = <array of mandatory non-key property names (see CIM
734         Requirements)>;
735 }
736 &smShowInstances ( "CIM_RecordLog", "CIM_MemberOfCollection",
737     $containerInstance.getInstancePath(), #propertylist[] );
738 &smEnd;

```

### 739 6.5.4.2 Show Command Form for a Single Instance Target

740 This command form is used to show a single instance of CIM\_RecordLog.

#### 741 6.5.4.2.1 Command Form

```
742 show <CIM_RecordLog single instance>
```

#### 743 6.5.4.2.2 CIM Requirements

744 See CIM\_RecordLog in the "CIM Elements" section of the [Record Log Profile](#) for the list of mandatory  
745 properties.

#### 746 6.5.4.2.3 Behavior Requirements

##### 747 6.5.4.2.3.1 Preconditions

748 In this section \$instance represents the targeted instance of CIM\_RecordLog.

```
749 $instance=<CIM_RecordLog single instance>;
```

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

##### 751 6.5.4.2.3.2 Pseudo Code

```

752 #propertylist[] = NULL;
753 if ( false == #all)
754 {
755     #propertylist[] = <array of mandatory non-key property names (see CIM
756         Requirements)>;
757 }
758 &smShowInstance ( $instance.getInstancePath(), #propertylist[] );
759 &smEnd;

```

### 760 6.5.5 Start

761 This section describes how to implement the `start` verb when applied to an instance of  
762 CIM\_RecordLog. Implementations may support the use of the `start` verb with CIM\_RecordLog.

#### 763 6.5.5.1 Command Form

```
764 start <CIM_RecordLog single instance>
```

#### 765 6.5.5.2 CIM Requirements

```
766 uint16 EnabledState;
```

```

767 uint16 RequestedState;
768 uint32 CIM_RecordLog.RequestStateChange (
769     [IN] uint16 RequestedState,
770     [OUT] REF CIM_ConcreteJob Job,
771     [IN] datetime TimeoutPeriod );

```

### 772 6.5.5.3 Behavior Requirements

#### 773 6.5.5.3.1.1 Preconditions

774 In this section `$instance` represents the targeted instance of `CIM_RecordLog`.

```
775 $instance=<CIM_RecordLog single instance>;
```

#### 776 6.5.5.3.1.2 Pseudo Code

```

777 &smStartRSC ( $instance.getInstancePath() );
778 &smEnd;

```

### 779 6.5.6 Stop

780 This section describes how to implement the `stop` verb when applied to an instance of `CIM_RecordLog`.  
781 Implementations may support the use of the `stop` verb with `CIM_RecordLog`.

#### 782 6.5.6.1 Command Form

```
783 stop <CIM_RecordLog single instance>
```

#### 784 6.5.6.2 CIM Requirements

```

785 uint16 EnabledState;
786 uint16 RequestedState;
787 uint32 CIM_RecordLog.RequestStateChange (
788     [IN] uint16 RequestedState,
789     [OUT] REF CIM_ConcreteJob Job,
790     [IN] datetime TimeoutPeriod );

```

### 791 6.5.6.3 Behavior Requirements

#### 792 6.5.6.3.1 Preconditions

793 In this section `$instance` represents the targeted instance of `CIM_RecordLog`.

```
794 $instance=<CIM_RecordLog single instance>;
```

#### 795 6.5.6.3.1.1 Pseudo Code

```

796 &smStopRSC ( $instance.getInstancePath() );
797 &smEnd;

```

### 798 6.6 CIM\_UseOfLog

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

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

805

**Table 6 – Command Verb Requirements for CIM\_UseOfLog**

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	

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

### 808 6.6.1 Ordering of Results

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

- 811 • Results for CIM\_UseOfLog are unordered; therefore, no algorithm is defined.

### 812 6.6.2 Show

813 This section describes how to implement the `show` verb when applied to an instance of CIM\_UseOfLog.  
814 Implementations shall support the use of the `show` verb with CIM\_UseOfLog.

#### 815 6.6.2.1 Show Command Form for Multiple Instances Target – CIM\_RecordLog Reference

816 This command form is used to show many instances of CIM\_UseOfLog. This command form corresponds  
817 to a `show` command issued against instances of CIM\_UseOfLog where only one reference is specified  
818 and the reference is to the scoping instance of CIM\_RecordLog.

##### 819 6.6.2.1.1 Command Form

```
820 show <CIM_UseOfLog multiple instances>
```

##### 821 6.6.2.1.2 CIM Requirements

822 See CIM\_UseOfLog in the “CIM Elements” section of the [Record Log Profile](#) for the list of mandatory  
823 properties.

##### 824 6.6.2.1.3 Behavior Requirements

###### 825 6.6.2.1.3.1 Preconditions

826 In this section `$instance` represents the instance of CIM\_RecordLog which is referenced by  
827 CIM\_UseOfLog.

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

829 **6.6.2.1.3.2 Pseudo Code**

```

830 #propertylist[] = NULL;
831 if ( false == #all)
832 {
833     #propertylist[] = <array of mandatory non-key property names (see CIM
834         Requirements)>;
835 }
836 &smShowAssociationInstances ( "CIM_UseOfLog", $instance.getInstancePath(),
837     #propertylist[] );
838 &smEnd;

```

839 **6.6.2.2 Show Command Form for Multiple Instances – CIM\_ManagedSystemElement Reference**

840 This command form is used to show many instances of CIM\_UseOfLog. This command form corresponds  
841 to a `show` command issued against instances of CIM\_UseOfLog where only one reference is specified  
842 and the reference is to the scoping instance of a subclass of CIM\_ManagedSystemElement.

843 **6.6.2.2.1 Command Form**

```
844 show <CIM_UseOfLog multiple instances>
```

845 **6.6.2.2.2 CIM Requirements**

846 See CIM\_UseOfLog in the “CIM Elements” section of the [Record Log Profile](#) for the list of mandatory  
847 properties.

848 **6.6.2.2.3 Behavior Requirements**849 **6.6.2.2.3.1 Preconditions**

850 In this section `$instance` represents the instance of a subclass of CIM\_ManagedSystemElement which  
851 is referenced by CIM\_UseOfLog.

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

853 **6.6.2.2.3.2 Pseudo Code**

```

854 #propertylist[] = NULL;
855 if ( false == #all)
856 {
857     #propertylist[] = <array of mandatory non-key property names (see CIM
858         Requirements)>;
859 }
860 &smShowAssociationInstances ( "CIM_UseOfLog", $instance.getInstancePath(),
861     propertylist[] );
862 &smEnd;

```

863 **6.6.2.3 Show Command Form for a Single Instance Target – Both References**

864 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
865 a `show` command issued against CIM\_UseOfLog where both references are specified and therefore the  
866 desired instance is unambiguously identified.

867 **6.6.2.3.1 Command Form**

```
868 show <CIM_UseOfLog single instance>
```

**869 6.6.2.3.2 CIM Requirements**

870 See CIM\_UseOfLog in the “CIM Elements” section of the [Record Log Profile](#) for the list of mandatory  
871 properties.

**872 6.6.2.3.3 Behavior Requirements****873 6.6.2.3.3.1 Preconditions**

874 In this section `$instanceA` represents the referenced instance of CIM\_RecordLog through  
875 CIM\_UseOfLog association. `$instanceB` represents the instance of a subclass of  
876 CIM\_ManagedSystemElement which is referenced by CIM\_UseOfLog.

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

**878 6.6.2.3.3.2 Pseudo Code**

```
879 #propertylist[] = NULL;  
880 if ( false == #all)  
881 {  
882     #propertylist[] = <array of mandatory non-key property names (see CIM  
883         Requirements)>;  
884 }  
885 &smShowAssociationInstance ( "CIM_UseOfLog", $instanceA.getInstancePath(),  
886     $instanceB.getInstancePath(), propertylist[] );  
887 &smEnd;
```

888

889  
890  
891  
892  
893

**ANNEX A**  
(informative)

**Change Log**

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

894