



1
2
3
4

Document Number: DSP0827

Date: 2009-07-14

Version: 1.0.0

5 **Software Update 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27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

CONTENTS

31	Foreword	5
32	Introduction	6
33	1 Scope	7
34	2 Normative References.....	7
35	2.1 Approved References	7
36	2.2 Other References.....	7
37	3 Terms and Definitions.....	7
38	4 Symbols and Abbreviated Terms.....	8
39	5 Recipes.....	9
40	5.1 smInstallFromURI	9
41	5.2 smInstallFromSoftwareIdentity	12
42	6 Mappings.....	15
43	6.1 CIM_SoftwareIdentity.....	15
44	6.2 CIM_HostedService	17
45	6.3 CIM_SoftwareInstallationServiceCapabilities	20
46	6.4 CIM_ElementCapabilities	22
47	6.5 CIM_ServiceAffectsElement	25
48	6.6 CIM_ManagedElement	31
49	ANNEX A (informative) Change Log	35

50

51 Tables

52	Table 1 – Command Verb Requirements for CIM_SoftwareInstallationService.....	15
53	Table 2 – Command Verb Requirements for CIM_HostedService	17
54	Table 3 – Command Verb Requirements for CIM_SoftwareInstallationServiceCapabilities	20
55	Table 4 – Command Verb Requirements for CIM_ElementCapabilities	22
56	Table 5 – Command Verb Requirements for CIM_ServiceAffectsElement	25
57	Table 6 – Command Verb Requirements for CIM_ManagedElement	31

58

60

Foreword

61 The *Software Update Profile SM CLP Mapping Specification* (DSP0827) was prepared by the Server
62 Management Working Group.

63 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems
64 management and interoperability.

65 **Conventions**

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

68 **Acknowledgements**

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

- 71 • RadhaKrishna R. Dasari – Dell
- 72 • Khachatur Papanyan – Dell
- 73 • Perry Vincent – Intel
- 74 • Aaron Merkin – IBM

75

76

Introduction

77 This document defines the SM CLP mapping for CIM elements described in the [Software Update Profile](#).
78 The information in this specification, combined with *SM CLP-to-CIM Common Mapping Specification 1.0*
79 ([DSP0216](#)), is intended to be sufficient to implement SM CLP commands relevant to the classes,
80 properties, and methods described in the [Software Update Profile](#) using CIM operations.

81 The target audience for this specification is implementers of the SM CLP support for the [Software Update](#)
82 [Profile](#).

83

84 Software Update Profile SM CLP Mapping Specification

85 1 Scope

86 This specification contains the requirements for an implementation of the SM CLP to provide access to
87 and implement the behaviors of the [Software Update Profile](#).

88 2 Normative References

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

92 2.1 Approved References

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

95 DMTF DSP1025, *Software Update Profile 1.0*,
96 http://www.dmtf.org/standards/published_documents/DSP1025_1.0.pdf

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

99 2.2 Other References

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

102 3 Terms and Definitions

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

104 3.1

105 **can**

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

107 3.2

108 **cannot**

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

110 3.3

111 **conditional**

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

114 3.4

115 **mandatory**

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

- 118 **3.5**
119 **may**
120 indicates a course of action permissible within the limits of the document
- 121 **3.6**
122 **need not**
123 indicates a course of action permissible within the limits of the document
- 124 **3.7**
125 **optional**
126 indicates a course of action permissible within the limits of the document
- 127 **3.8**
128 **shall**
129 indicates requirements to be followed strictly in order to conform to the document and from which no
130 deviation is permitted
- 131 **3.9**
132 **shall not**
133 indicates requirements to be followed strictly in order to conform to the document and from which no
134 deviation is permitted
- 135 **3.10**
136 **should**
137 indicates that among several possibilities, one is recommended as particularly suitable, without
138 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 139 **3.11**
140 **should not**
141 indicates that a certain possibility or course of action is deprecated but not prohibited

142 **4 Symbols and Abbreviated Terms**

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

- 144 **4.1**
145 **CIM**
146 Common Information Model
- 147 **4.2**
148 **CLP**
149 Command Line Protocol
- 150 **4.3**
151 **DMTF**
152 Distributed Management Task Force
- 153 **4.4**
154 **SM**
155 Server Management

- 156 **4.5**
 157 **SMI-S**
 158 Storage Management Initiative Specification
- 159 **4.6**
 160 **SNIA**
 161 Storage Networking Industry Association
- 162 **4.7**
 163 **UFsT**
 164 User Friendly selection Tag

165 **5 Recipes**

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

- 168 • smShowInstance
- 169 • smShowInstances
- 170 • smShowAssociationInstance
- 171 • smShowAssociationInstances
- 172 • smNewInstance
- 173 • smAddError

174 There are no Local Recipes defined for use in this mapping.

175 **5.1 smInstallFromURI**

176 **5.1.1 Description**

177 This method is used to install software that is located at a URI on a managed element.

178 **5.1.2 Pseudo Code**

```

179 sub void smInstallfromURI (#InstallOptions[],#uri, $targetME->, $targetSWInsSer->)
180 {
181 // #InstallOptions contains the list of options that control the installation procedure
182 // $targetME parameter contains the target managed element on which the software needs
183 // to be installed. uri parameter contains the URI at which the software is located
184 $instanceConcreteJob = smNewInstance ("CIM_ConcreteJob");
185 %InArguments[] = { newArgument("InstallOptions[]", #InstallOptions),
186   newArgument("URI", #uri), newArgument("Target", $targetME->) }
187 %OutArguments[] = { newArgument("Job", $instanceConcreteJob.getObjectPath() ) }
188 #Error = smOpInvokeMethod ($targetSWInsSer->,
189   "InstallFromURI",
190   %InArguments[],
191   %OutArguments[],
192   #returnStatus);
193 if (0 != #Error.code)
194 {
195 //method invocation failed

```

```

196 if ( (null != #Error.$error) && (null != #Error.$error[0]) )
197 {
198     //if the method invocation contains an embedded error
199     //use it for the Error for the overall job
200     &smAddError($job, #Error.$error[0]);
201     &smMakeCommandStatus($job);
202     &smEnd;
203 }
204     else if (17 == #returnStatus) {
205         //The specified extrinsic method does not exist
206         $OperationError = smNewInstance("CIM_Error");
207         //CIM_ERR_METHOD_NOT_FOUND
208         $OperationError.CIMStatusCode = 17;
209         //Software Error
210         $OperationError.ErrorType = 10;
211         //Low
212         $OperationError.PerceivedSeverity = 0;
213         $OperationError.OwningEntity = DMTF:SMCLP;
214             $OperationError.MessageID = 0x00000001;
215             $OperationError.Message = "Operation is not supported";
216             &smAddError($job, $OperationError);
217         &smMakeCommandStatus($job);
218     }
219     else
220     {
221         //operation failed, but no detailed error instance, need to make //one up
222         //make an Error instance and associate with job for Operation
223         $OperationError = smNewInstance("CIM_Error");
224         //CIM_ERR_FAILED
225         $OperationError.CIMStatusCode = 1;
226         //Software Error
227         $OperationError.ErrorType = 4;
228         //Unknown
229         $OperationError.PerceivedSeverity = 0;
230         $OperationError.OwningEntity = DMTF:SMCLP;
231         $OperationError.MessageID = 0x00000009;
232         $OperationError.Message = "An internal software error has occurred.";
233         &smAddError($job, $OperationError);
234         &smMakeCommandStatus($job);
235         &smEnd;
236     }
237 } //if CIM op failed
238 else if (0 == #returnStatus) {
239     //completed successfully
240     &smCommandCompleted($job);
241     &smEnd;
242 }
243     else if (4096 == #returnStatus) {
244         //job spawned, need to watch for it to finish
245         //while the jobstate is 4 ("Running")
246         while (4 == $instanceConcreteJob.JobState){<busy wait>}
247         //when job finishes, invoke GetError()

```

```

248 if (2 != $job.OperationalStatus) {
249 %InArguments[] = { }
250 %OutArguments[] = {newArgument("Job", $instanceConcreteJob.getObjectPath())}
251 #Error = smOpInvokeMethod($job, "GetError", %InArguments, %OutArguments, #returncode);
252 //Method invocation failed, internal processing error
253 if ( (0 != #Error.code) || (0 != #returncode) ) {
254 //make an Error instance and associate with job for Operation
255 $OperationError = smNewInstance("CIM_Error");
256 //CIM_ERR_FAILED
257 $OperationError.CIMStatusCode = 1;
258 //Software Error
259 $OperationError.ErrorType = 4;
260 //Unknown
261 $OperationError.PerceivedSeverity = 0;
262 $OperationError.OwningEntity = DMTF:SMCLP;
263 $OperationError.MessageID = 0x00000009;
264 $OperationError.Message = "An internal software error has occurred.";
265 &smAddError($job, $OperationError);
266 &smMakeCommandStatus($job);
267 &smEnd;
268 } else {
269 //make command status
270 $joberror = %OutArguments["Error"];
271 &smMakeCommandExecutionFailed($job, {$joberror});
272 } //end if have CIM_Error from GetError()
273 }
274 else if (1 == #returnStatus) {
275 //unsupported
276 $OperationError = smNewInstance("CIM_Error");
277 //CIM_ERR_NOT_SUPPORTED
278 $OperationError.CIMStatusCode = 7;
279 //Other
280 $OperationError.ErrorType = 1;
281 //Low
282 $OperationError.PerceivedSeverity = 2;
283 $OperationError.OwningEntity = DMTF:SMCLP;
284 $OperationError.MessageID = 0x00000001;
285 $OperationError.Message = "Operation is not supported.";
286 &smAddError($job, $OperationError);
287 &smMakeCommandStatus($job);
288 &smEnd;
289 }
290 else if (2 == #returnStatus) {
291 //generic failure
292 $OperationError = smNewInstance("CIM_Error");
293 //CIM_ERR_FAILED
294 $OperationError.CIMStatusCode = 1;
295 //Other
296 $OperationError.ErrorType = 1;
297 //Low
298 $OperationError.PerceivedSeverity = 2;
299 $OperationError.OwningEntity = DMTF:SMCLP;
300 $OperationError.MessageID = 0x00000002;

```

```

301     $OperationError.Message = "Failed. No further information is available.";
302     &smAddError($job, $OperationError);
303     &smMakeCommandStatus($job);
304 }
305 else {
306     //unspecified return code, generic failure
307     $OperationError = smNewInstance("CIM_Error");
308     //CIM_ERR_FAILED
309     $OperationError.CIMStatusCode = 1;
310     //Other
311     $OperationError.ErrorType = 1;
312     //Low
313     $OperationError.PerceivedSeverity = 2;
314     $OperationError.OwningEntity = DMTF:SMCLP;
315     $OperationError.MessageID = 0x00000002;
316     $OperationError.Message = "Failed. No further information is available.";
317     &smAddError($job, $OperationError);
318     &smMakeCommandStatus($job);
319     &smEnd;
320 }

```

321 5.2 smInstallFromSoftwareIdentity

322 5.2.1 Description

323 This method is used to install software, represented by an instance of CIM_SoftwareIdentity, on a
 324 managed element.

325 5.2.2 Pseudo Code

```

326 sub void smInstallFromSoftwareIdentity (#InstallOptions[], $SWInstance->, $targetME->,
327     $targetSWInsSer->)
328 {
329     //InstallOptions[] contains the list of options that control the installation
330     procedure
331     // $SWInstance represents the instance of CIM_SoftwareIdentity
332     // $targetME parameter contains the target managed element on which the software needs
333     to be installed.
334     $instanceConcreteJob = smNewInstance ("CIM_ConcreteJob");
335     %InArguments[] = {newArgument("InstallOptions[]", #InstallOptions),
336     newArgument("Source", $SWInstance->),
337     newArgument("Target", $targetME->)}
338     %OutArguments[] = {newArgument("Job", $instanceConcreteJob.getObjectPath())}
339     #Error = smOpInvokeMethod ($targetSWInsSer->,
340     "InstallFromSoftwareIdentity",
341     %InArguments[],
342     %OutArguments[],
343     #returnStatus);
344     if (0 != #Error.code)
345     {
346     //method invocation failed
347     if ( (null != #Error.$error) && (null != #Error.$error[0]) )
348     {
349     //if the method invocation contains an embedded error

```

```

350 //use it for the Error for the overall job
351 &smAddError($job, #Error.$error[0]);
352 &smMakeCommandStatus($job);
353 &smEnd;
354 }
355     else if (17 == #returnStatus) {
356         //The specified extrinsic method does not exist
357 $OperationError = smNewInstance("CIM_Error");
358 //CIM_ERR_METHOD_NOT_FOUND
359 $OperationError.CIMStatusCode = 17;
360 //Software Error
361 $OperationError.ErrorType = 10;
362 //Low
363 $OperationError.PerceivedSeverity = 0;
364 $OperationError.OwningEntity = DMTF:SMCLP;
365     $OperationError.MessageID = 0x00000001;
366     $OperationError.Message = "Operation is not supported";
367     &smAddError($job, $OperationError);
368     &smMakeCommandStatus($job);
369     }
370     else
371     {
372 //operation failed, but no detailed error instance, need to make //one up
373 //make an Error instance and associate with job for Operation
374     $OperationError = smNewInstance("CIM_Error");
375     //CIM_ERR_FAILED
376     $OperationError.CIMStatusCode = 1;
377     //Software Error
378     $OperationError.ErrorType = 4;
379     //Unknown
380     $OperationError.PerceivedSeverity = 0;
381     $OperationError.OwningEntity = DMTF:SMCLP;
382     $OperationError.MessageID = 0x00000009;
383 $OperationError.Message = "An internal software error has occurred.";
384     &smAddError($job, $OperationError);
385     &smMakeCommandStatus($job);
386     &smEnd;
387     }
388 }//if CIM op failed
389 else if (0 == #returnStatus) {
390     //completed successfully
391     &smCommandCompleted($job);
392     &smEnd;
393 }
394 else if (4096 == #returnStatus) {
395 //job spawned, need to watch for it to finish
396 //while the jobstate is 4 ("Running")
397 while (4 == $instanceConcreteJob.JobState){<busy wait>}
398 //when job finishes, invoke GetError()
399 if (2 != $job.OperationalStatus) {
400 %InArguments[] = { }
401 %OutArguments[] = {newArgument("Job", $instanceConcreteJob.getObjectPath())}
402 #Error = smOpInvokeMethod($job, "GetError", %InArguments, %OutArguments, #returncode);

```

```

403 //Method invocation failed, internal processing error
404 if ( (0 != #Error.code) || (0 != #returncode) ) {
405 //make an Error instance and associate with job for Operation
406 $OperationError = smNewInstance("CIM_Error");
407 //CIM_ERR_FAILED
408 $OperationError.CIMStatusCode = 1;
409 //Software Error
410 $OperationError.ErrorType = 4;
411 //Unknown
412 $OperationError.PerceivedSeverity = 0;
413 $OperationError.OwningEntity = DMTF:SMCLP;
414 $OperationError.MessageID = 0x00000009;
415 $OperationError.Message = "An internal software error has occurred.";
416 &smAddError($job, $OperationError);
417 &smMakeCommandStatus($job);
418 &smEnd;
419 } else {
420 //make command status
421 $joberror = %OutArguments["Error"];
422 &smMakeCommandExecutionFailed($job, {$joberror});
423 } //end if have CIM_Error from GetError()
424 }
425 else if (1 == #returnStatus) {
426 //unsupported
427 $OperationError = smNewInstance("CIM_Error");
428 //CIM_ERR_NOT_SUPPORTED
429 $OperationError.CIMStatusCode = 7;
430 //Other
431 $OperationError.ErrorType = 1;
432 //Low
433 $OperationError.PerceivedSeverity = 2;
434 $OperationError.OwningEntity = DMTF:SMCLP;
435 $OperationError.MessageID = 0x00000001;
436 $OperationError.Message = "Operation is not supported.";
437 &smAddError($job, $OperationError);
438 &smMakeCommandStatus($job);
439 &smEnd;
440 }
441 else if (2 == #returnStatus) {
442 //generic failure
443 $OperationError = smNewInstance("CIM_Error");
444 //CIM_ERR_FAILED
445 $OperationError.CIMStatusCode = 1;
446 //Other
447 $OperationError.ErrorType = 1;
448 //Low
449 $OperationError.PerceivedSeverity = 2;
450 $OperationError.OwningEntity = DMTF:SMCLP;
451 $OperationError.MessageID = 0x00000002;
452 $OperationError.Message = "Failed. No further information is available.";
453 &smAddError($job, $OperationError);
454 &smMakeCommandStatus($job);
455 }

```

```

456 else {
457     //unspecified return code, generic failure
458     $OperationError = smNewInstance("CIM_Error");
459     //CIM_ERR_FAILED
460     $OperationError.CIMStatusCode = 1;
461     //Other
462     $OperationError.ErrorType = 1;
463     //Low
464     $OperationError.PerceivedSeverity = 2;
465     $OperationError.OwningEntity = DMTF:SMCLP;
466     $OperationError.MessageID = 0x00000002;
467     $OperationError.Message = "Failed. No further information is available.";
468     &smAddError($job, $OperationError);
469     &smMakeCommandStatus($job);
470     &smEnd;
471 }

```

472 6 Mappings

473 The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in
 474 the [Software Update Profile](#). Requirements specified here related to support for a CLP verb for a
 475 particular class are solely within the context of this profile.

476 6.1 CIM_SoftwareIdentity

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

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

483 **Table 1 – Command Verb Requirements for CIM_SoftwareInstallationService**

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 6.1.2.
start	Not supported	
stop	Not supported	

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

486 6.1.1 Ordering of Results

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

- 489 • Results for CIM_SoftwareInstallationService are unordered; therefore, no algorithm is defined.

490 6.1.2 Show

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

494 6.1.2.1 Show Command Form for Single Object Target – CIM_SoftwareInstallationService

495 This command form is used to show many instances of CIM_SoftwareInstallationService. This command
496 form corresponds to a `show` command issued against instances of CIM_SoftwareInstallationService
497 where only one reference is specified and the reference is to an instance of
498 CIM_SoftwareInstallationService.

499 6.1.2.1.1 Command Form

```
500 show <CIM_SoftwareInstallationService single object>
```

501 6.1.2.1.2 CIM Requirements

502 See CIM_SoftwareInstallationService in the “CIM Elements” section of the [Software Update Profile](#) for the
503 list of mandatory properties.

504 6.1.2.1.3 Behavior Requirements

505 6.1.2.1.3.1 Preconditions

506 `$instance` represents the targeted instance of CIM_SoftwareInstallationService.

```
507 $instance = <CIM_SoftwareInstallationService single object>;
```

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

509 6.1.2.1.3.2 Pseudo Code

```
510 #propertylist[] = NULL;
511 if ( false == #all )
512 {
513     #propertylist[] = {<array of mandatory non-key property names (see CIM
514         Requirements)>}
515 }
516 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
517 &smEnd;
```


518 **6.1.2.2 Show Command Form for Multiple Objects Target – CIM_SoftwareInstallationService**

519 **6.1.2.2.1 Command Form**

520 `show <CIM_SoftwareInstallationService multiple objects>`

521 **6.1.2.2.2 CIM Requirements**

522 See CIM_SoftwareInstallationService in the “CIM Elements” section of the [Software Update Profile](#) for the
523 list of mandatory properties.

524 **6.1.2.2.3 Behavior Requirements**

525 **6.1.2.2.3.1 Preconditions**

526 \$containerInstance is the instance of CIM_ComputerSystem or CIM_System that is associated with
527 the targeted instances of CIM_SoftwareInstallationService through the CIM_HostedService association.

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

529 **6.1.2.2.3.2 Pseudo Code**

```
530 #propertylist[] = NULL;
531 if ( false == #all )
532 {
533     #propertylist[] = {<array of mandatory non-key property names (see CIM
534         Requirements)>}
535 }
536 &smShowInstances ( "CIM_SoftwareInstallationService", "CIM_HostedService",
537     $containerInstance.getObjectPath(), #propertylist[] );
538 &smEnd;
```

539 **6.2 CIM_HostedService**

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

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

546 **Table 2 – 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 6.2.2.
start	Not supported	

Command Verb	Requirement	Comments
stop	Not supported	

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

549 6.2.1 Ordering of Results

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

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

553 6.2.2 Show

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

556 6.2.2.1 Show Command Form for Multiple Instances Target – CIM_ComputerSystem Reference

557 This command form is used to show many instances of CIM_HostedService. This command form
558 corresponds to a `show` command issued against instances of CIM_HostedService where only one
559 reference is specified and the reference is to the scoping instance of CIM_ComputerSystem.

560 6.2.2.1.1 Command Form

561 `show <CIM_HostedService multiple instances>`

562 6.2.2.1.2 CIM Requirements

563 See CIM_HostedService in the “CIM Elements” section of the [Software Update Profile](#) for the list of
564 mandatory properties.

565 6.2.2.1.3 Behavior Requirements

566 6.2.2.1.3.1 Preconditions

567 `$instance` represents the instance of a CIM_ComputerSystem, which is referenced by
568 CIM_HostedService.

569 `$instance=<CIM_ComputerSystem single instance>;`

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

571 6.2.2.1.3.2 Pseudo Code

```
572 #propertylist[] = NULL;
573 if ( false == #all )
574 {
575     #propertylist[] = <array of mandatory non-key property names (see CIM
576     Requirements)>;
577 }
578 &smShowAssociationInstances ( "CIM_HostedService", $instance.getObjectPath(),
579     #propertylist[] );
580 &smEnd;
```

581 **6.2.2.2 Show Command Form for a Single Instance – CIM_SoftwareInstallationService Reference**

582 This command form is used to show a single instance of CIM_HostedService. This command form
583 corresponds to a `show` command issued against a single instance of CIM_HostedService where only one
584 reference is specified and the reference is to the instance of CIM_SoftwareInstallationService.

585 **6.2.2.2.1 Command Form**

```
586 show <CIM_HostedService single instance>
```

587 **6.2.2.2.2 CIM Requirements**

588 See CIM_HostedService in the “CIM Elements” section of the [Software Update Profile](#) for the list of
589 mandatory properties.

590 **6.2.2.2.3 Behavior Requirements**

591 **6.2.2.2.3.1 Preconditions**

592 `$instance` represents the instance of a CIM_SoftwareInstallationService, which is referenced by
593 CIM_HostedService.

```
594 $instance=<CIM_SoftwareInstallationService single instance>
```

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

596 **6.2.2.2.3.2 Pseudo Code**

```
597 #propertylist[] = NULL;
598 if ( false == #all )
599     {
600         #propertylist[] = <array of mandatory non-key property names (see CIM
601             Requirements)>;
602     }
603 &smShowAssociationInstances ( "CIM_HostedService", $instance.getObjectPath(),
604     #propertylist[] );
605 &smEnd;
```

606 **6.2.2.3 Show Command Form for a Single Instance Target – Both References**

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

610 **6.2.2.3.1 Command Form**

```
611 show <CIM_HostedService single instance>
```

612 **6.2.2.3.2 CIM Requirements**

613 See CIM_HostedService in the “CIM Elements” section of the [Software Update Profile](#) for the list of
614 mandatory properties.

615 **6.2.2.3.3 Behavior Requirements**616 **6.2.2.3.3.1 Preconditions**

617 In this section `$instanceA` represents the instance of a `CIM_ComputerSystem` and `$instanceB`
 618 represents the instance of `CIM_SoftwareInstallationService`, both of which are referenced by
 619 `CIM_HostedService`.

```
620 $instanceA=<CIM_ComputerSystem single instance>;
621 $instanceB=<CIM_SoftwareInstallationService single instance>;
```

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

623 **6.2.2.3.3.2 Pseudo Code**

```
624 #propertylist[] = NULL;
625 if ( false == #all )
626 {
627     #propertylist[] = <array of mandatory non-key property names (see CIM
628         Requirements)>;
629 }
630 &smShowAssociationInstance ( "CIM_HostedService", $instanceA.getObjectPath(),
631     $instanceB.getObjectPath(), #propertylist[] );
632 &smEnd;
```

633 **6.3 CIM_SoftwareInstallationServiceCapabilities**

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

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

640 **Table 3 – Command Verb Requirements for CIM_SoftwareInstallationServiceCapabilities**

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	

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

643 **6.3.1 Ordering of Results**

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

- 646 • Results for CIM_SoftwareInstallationServiceCapabilities are unordered; therefore, no algorithm
647 is defined.

648 **6.3.2 Show**

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

652 The `show` verb is used to display information about CIM_SoftwareInstallationServiceCapabilities
653 instances.

654 **6.3.2.1 Show Command Form for a Single Instance Target**

655 **6.3.2.1.1 Command Form**

```
656 show <CIM_SoftwareInstallationServiceCapabilities single instance>
```

657 **6.3.2.1.2 CIM Requirements**

658 See CIM_SoftwareInstallationServiceCapabilities in the “CIM Elements” section of the [Software Update](#)
659 [Profile](#) for the list of mandatory properties.

660 **6.3.2.1.3 Behavior Requirements**

661 **6.3.2.1.3.1 Preconditions**

662 `$instance` represents the targeted instance of CIM_SoftwareInstallationServiceCapabilities.

```
663 $instance=<CIM_SoftwareInstallationServiceCapabilities single instance>;
```

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

665 **6.3.2.1.3.2 Pseudo Code**

```
666 #propertylist[] = NULL;
667 if ( false == #all )
668 {
669     #propertylist[] = {<array of mandatory non-key property names (see CIM
670     Requirements)>}
671 }
672 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
673 &smEnd;
```

674 **6.3.2.2 Show Command Form for Multiple Instances Target**

675 **6.3.2.2.1 Command Form**

676 `show <CIM_SoftwareInstallationServiceCapabilities multiple instances>`

677 **6.3.2.2.2 CIM Requirements**

678 See CIM_SoftwareInstallationServiceCapabilities in the “CIM Elements” section of the [Software Update Profile](#) for the list of mandatory properties.

680 **6.3.2.2.3 Behavior Requirements**

681 **6.3.2.2.3.1 Preconditions**

682 \$containerInstance represents the instance of CIM_ConcreteCollection with ElementName property
683 that contains “Capabilities” and is associated with the targeted instances of
684 CIM_SoftwareInstallationServiceCapabilities through the CIM_MemberOfCollection association.

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

686 **6.3.2.2.3.2 Pseudo Code**

```
687 #propertylist[] = NULL;
688 if ( false == #all )
689     {
690         #propertylist[] = {<array of mandatory non-key property names (see CIM
691             Requirements)>}
692     }
693 &smShowInstances ( "CIM_SoftwareInstallationServiceCapabilities",
694     "CIM_MemberOfCollection", $containerInstance.getObjectPath(), #propertylist[] );
695 &smEnd;
```

696 **6.4 CIM_ElementCapabilities**

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

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

703 **Table 4 – 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 6.4.2.

Command Verb	Requirement	Comments
start	Not supported	
stop	Not supported	

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

706 6.4.1 Ordering of Results

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

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

710 6.4.2 Show

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

714 6.4.2.1 Show Command Form for a Single Instance Target – CIM_SoftwareInstallationService 715 Reference

716 This command form is used to show a single instance of CIM_ElementCapabilities. This command form
717 corresponds to a `show` command issued against an instance of CIM_ElementCapabilities where only one
718 reference is specified and the reference is to the instance of CIM_SoftwareInstallationService.

719 6.4.2.1.1 Command Form

```
720 show <CIM_ElementCapabilities multiple instances>
```

721 6.4.2.1.2 CIM Requirements

722 See CIM_ElementCapabilities in the “CIM Elements” section of the [Software Update Profile](#) for the list of
723 mandatory properties.

724 6.4.2.1.3 Behavior Requirements

725 6.4.2.1.3.1 Preconditions

726 `$instance` represents the instance of a CIM_SoftwareInstallationService, which is referenced by
727 CIM_ElementCapabilities.

```
728 $instance=<CIM_SoftwareInstallationService single instance>;
```

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

730 6.4.2.1.3.2 Pseudo Code

```
731 #propertylist[] = NULL;
732 if ( false == #all )
733 {
734     #propertylist[] = <array of mandatory non-key property names (see CIM
735     Requirements)>;
736 }
737 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath(),
```

```
738     #propertylist[] );
739 &smEnd;
```

740 6.4.2.2 Show Command Form for Multiple Instances –

741 CIM_SoftwareInstallationServiceCapabilities Reference

742 This command form is used to show multiple instances of CIM_ElementCapabilities. This command form
743 corresponds to a `show` command issued against CIM_ElementCapabilities where only one reference is
744 specified and the reference is to the instance of CIM_SoftwareInstallationServiceCapabilities.

745 6.4.2.2.1 Command Form

```
746 show <CIM_ElementCapabilities multiple instances>
```

747 6.4.2.2.2 CIM Requirements

748 See CIM_ElementCapabilities in the “CIM Elements” section of the [Software Update Profile](#) for the list of
749 mandatory properties.

750 6.4.2.2.3 Behavior Requirements

751 6.4.2.2.3.1 Preconditions

752 `$instance` represents the instance of a CIM_SoftwareInstallationServiceCapabilities, which is
753 referenced by CIM_ElementCapabilities.

```
754 $instance=<CIM_SoftwareInstallationServiceCapabilities single instance>.
```

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

756 6.4.2.2.3.2 Pseudo Code

```
757 #propertylist[] = NULL;
758 if ( false == #all )
759     {
760         #propertylist[] = <array of mandatory non-key property names (see CIM
761             Requirements)>;
762     }
763 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath(),
764     #propertylist[] );
765 &smEnd;
```

766 6.4.2.3 Show Command Form for a Single Instance Target – CIM_SoftwareInstallationService and

767 CIM_SoftwareInstallationServiceCapabilities References

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

771 6.4.2.3.1 Command Form

```
772 show <CIM_ElementCapabilities single instance>
```

773 6.4.2.3.2 CIM Requirements

774 See CIM_ElementCapabilities in the “CIM Elements” section of the [Software Update Profile](#) for the list of
775 mandatory properties.

776 **6.4.2.3.3 Behavior Requirements**

777 **6.4.2.3.3.1 Preconditions**

778 \$instanceA represents the instance of a CIM_SoftwareInstallationService and \$instanceB represents
 779 the instance of a CIM_SoftwareInstallationServiceCapabilities, both of which are referenced by
 780 CIM_ElementCapabilities.

```
781 $instanceA=<CIM_SoftwareInstallationService single instance>;
782 $instanceB=<CIM_SoftwareInstallationServiceCapabilities single instance>;
```

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

784 **6.4.2.3.3.2 Pseudo Code**

```
785 #propertylist[] = NULL;
786 if ( false == #all )
787 {
788     #propertylist[] = <array of mandatory non-key property names (see CIM
789         Requirements)>;
790 }
791 &smShowAssociationInstance ( "CIM_ElementCapabilities", $instanceA.getObjectPath(),
792     $instanceB.getObjectPath(), #propertylist[] );
793 &smEnd;
```

794 **6.5 CIM_ServiceAffectsElement**

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

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

801 **Table 5 – Command Verb Requirements for CIM_ServiceAffectsElement**

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

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

804 6.5.1 Ordering of Results

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

807 Results for CIM_ServiceAffectsElement are unordered; therefore, no algorithm is defined.

808 6.5.2 Show

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

812 6.5.2.1 Show Command Form for Multiple Instances Target – CIM_SoftwareInstallationService 813 Reference

814 This command form is used to show many instances of CIM_ServiceAffectsElement. This command form
815 corresponds to a `show` command issued against instances of CIM_ServiceAffectsElement where only
816 one reference is specified and the reference is to the scoping instance of
817 CIM_SoftwareInstallationService.

818 6.5.2.1.1 Command Form

```
819 show <CIM_ServiceAffectsElement multiple instances>
```

820 6.5.2.1.2 CIM Requirements

821 See CIM_ServiceAffectsElement in the “CIM Elements” section of the [Software Update Profile](#) for the list
822 of mandatory properties.

823 6.5.2.1.3 Behavior Requirements

824 6.5.2.1.3.1 Preconditions

825 `$instance` represents the instance of a CIM_SoftwareInstallationService, which is referenced by
826 CIM_ServiceAffectsElement.

```
827 $instance=<CIM_SoftwareInstallationService single instance>;
```

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

829 6.5.2.1.3.2 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_ServiceAffectsElement", $instance.getObjectPath(),
837     #propertylist[] );
838 &smEnd;
```

839 6.5.2.2 Show Command Form for Multiple Instances – CIM_SoftwareIdentity Reference

840 This command form is used to show multiple instances of CIM_ServiceAffectsElement. This command
841 form corresponds to a `show` command issued against multiple instances of CIM_ServiceAffectsElement
842 where only one reference is specified and the reference is to the instance of CIM_SoftwareIdentity.

843 **6.5.2.2.1 Command Form**844 `show <CIM_ServiceAffectsElement multiple instances>`845 **6.5.2.2.2 CIM Requirements**846 See CIM_ServiceAffectsElement in the “CIM Elements” section of the [Software Update Profile](#) for the list
847 of mandatory properties.848 **6.5.2.2.3 Behavior Requirements**849 **6.5.2.2.3.1 Preconditions**850 \$instance represents the instance of a CIM_SoftwareIdentity, which is referenced by
851 CIM_ServiceAffectsElement.852 `$instance=<CIM_SoftwareIdentity single instance>;`

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

854 **6.5.2.2.3.2 Pseudo Code**855 `#propertylist[] = NULL;`
856 `if (false == #all)`
857 `{`
858 `#propertylist[] = <array of mandatory non-key property names (see CIM`
859 `Requirements)>;`
860 `}`
861 `&smShowAssociationInstances ("CIM_ServiceAffectsElement", $instance.getObjectPath(),`
862 `#propertylist[]);`
863 `&smEnd;`864 **6.5.2.3 Show Command Form for a Single Instance Target – CIM_SoftwareInstallationService and**
865 **CIM_SoftwareIdentity References**866 This command form is for the show verb applied to a single instance. This command form corresponds to
867 the show command issued against CIM_ServiceAffectsElement where both references of
868 CIM_SoftwareInstallationService and CIM_SoftwareIdentity are specified and therefore the desired
869 instance is unambiguously identified.870 **6.5.2.3.1 Command Form**871 `show <CIM_ServiceAffectsElement single instance>`872 **6.5.2.3.2 CIM Requirements**873 See CIM_ServiceAffectsElement in the “CIM Elements” section of the [Software Update Profile](#) for the list
874 of mandatory properties.875 **6.5.2.3.3 Behavior Requirements**876 **6.5.2.3.3.1 Preconditions**877 \$instanceA represents the instance of a CIM_SoftwareInstallationService and \$instanceB represents
878 the instance of a CIM_SoftwareIdentity, both of which are referenced by CIM_ServiceAffectsElement.879 `$instanceA=<CIM_SoftwareInstallationService single instance>;`880 `$instanceB=<CIM_SoftwareIdentity single instance>;`

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

882 6.5.2.3.2 Pseudo Code

```
883 #propertylist[] = NULL;
884 if ( false == #all )
885 {
886     #propertylist[] = <array of mandatory non-key property names (see CIM
887         Requirements)>;
888 }
889 &smShowAssociationInstance ( "CIM_ServiceAffectsElement", $instanceA.getObjectPath(),
890     $instanceB.getObjectPath(), #propertylist[] );
891 &smEnd;
```

892 6.5.2.4 Show Command Form for Multiple Instances – CIM_ManagedElement Reference

893 This command form is used to show a single instance of CIM_ServiceAffectsElement. This command
894 form corresponds to a show command issued against multiple instances of CIM_ServiceAffectsElement
895 where only one reference is specified and the reference is to the instance of CIM_ManagedElement.

896 6.5.2.4.1 Command Form

```
897 show <CIM_ServiceAffectsElement multiple instances>
```

898 6.5.2.4.2 CIM Requirements

899 See CIM_ServiceAffectsElement in the “CIM Elements” section of the [Software Update Profile](#) for the list
900 of mandatory properties.

901 6.5.2.4.3 Behavior Requirements

902 6.5.2.4.3.1 Preconditions

903 \$instance represents the instance of a CIM_ManagedElement, which is referenced by
904 CIM_ServiceAffectsElement.

```
905 $instance=<CIM_ManagedElement single instance>;
```

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

907 6.5.2.4.3.2 Pseudo Code

```
908 #propertylist[] = NULL;
909 if ( false == #all )
910 {
911     #propertylist[] = <array of mandatory non-key property names (see CIM
912         Requirements)>;
913 }
914 &smShowAssociationInstances ( "CIM_ServiceAffectsElement", $instance.getObjectPath(),
915     #propertylist[] );
916 &smEnd;
```

917 6.5.2.5 Show Command Form for a Single Instance Target – CIM_SoftwareInstallationService and 918 CIM_ManagedElement References

919 This command form is for the show verb applied to a single instance. This command form corresponds to
920 the show command issued against CIM_ServiceAffectsElement where both references of

921 CIM_SoftwareInstallationService and CIM_ManagedElement are specified and therefore the desired
922 instance is unambiguously identified.

923 6.5.2.5.1 Command Form

```
924 show <CIM_ServiceAffectsElement single instance>
```

925 6.5.2.5.2 CIM Requirements

926 See CIM_ServiceAffectsElement in the “CIM Elements” section of the [Software Update Profile](#) for the list
927 of mandatory properties.

928 6.5.2.5.3 Behavior Requirements

929 6.5.2.5.3.1 Preconditions

930 \$instanceA represents the instance of a CIM_SoftwareInstallationService and \$instanceB represents
931 the instance of a CIM_ManagedElement, both of which are referenced by CIM_ServiceAffectsElement.

```
932 $instanceA=<CIM_SoftwareInstallationService single instance>;  
933 $instanceB=<CIM_ManagedElement single instance>;
```

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

935 6.5.2.5.3.2 Pseudo Code

```
936 #propertylist[] = NULL;  
937 if ( false == #all )  
938 {  
939     #propertylist[] = <array of mandatory non-key property names (see CIM  
940         Requirements)>;  
941 }  
942 &smShowAssociationInstance ( "CIM_ServiceAffectsElement", $instanceA.getObjectPath(),  
943     $instanceB.getObjectPath(), #propertylist[] );  
944 &smEnd;
```

945 6.5.2.6 Show Command Form for Multiple Instances – CIM_ComputerSystem Reference

946 This command form is used to show multiple instances of CIM_ServiceAffectsElement. This command
947 form corresponds to a show command issued against multiple instances of CIM_ServiceAffectsElement
948 where only one reference is specified and the reference is to the instance of CIM_ComputerSystem.

949 6.5.2.6.1 Command Form

```
950 show <CIM_ServiceAffectsElement multiple instances>
```

951 6.5.2.6.2 CIM Requirements

952 See CIM_ServiceAffectsElement in the “CIM Elements” section of the [Software Update Profile](#) for the list
953 of mandatory properties.

954 6.5.2.6.3 Behavior Requirements

955 6.5.2.6.3.1 Preconditions

956 \$instance represents the instance of a CIM_ComputerSystem, which is referenced by
957 CIM_ServiceAffectsElement.

```
958 $instance=<CIM_ComputerSystem single instance>;
```

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

960 6.5.2.6.3.2 Pseudo Code

```
961 #propertylist[] = NULL;
962 if ( false == #all )
963 {
964     #propertylist[] = <array of mandatory non-key property names (see CIM
965     Requirements)>;
966 }
967 &smShowAssociationInstances ( "CIM_ServiceAffectsElement", $instance.getObjectPath(),
968     #propertylist[] );
969 &smEnd;
```

970 6.5.2.7 Show Command Form for a Single Instance Target – CIM_SoftwareInstallationService and 971 CIM_ComputerSystem References

972 This command form is for the `show` verb applied to a single instance. This command form corresponds to
973 the `show` command issued against `CIM_ServiceAffectsElement` where both references of
974 `CIM_SoftwareInstallationService` and `CIM_ComputerSystem` are specified and therefore the desired
975 instance is unambiguously identified.

976 6.5.2.7.1 Command Form

```
977 show <CIM_ServiceAffectsElement single instance>
```

978 6.5.2.7.2 CIM Requirements

979 See `CIM_ServiceAffectsElement` in the “CIM Elements” section of the [Software Update Profile](#) for the list
980 of mandatory properties.

981 6.5.2.7.3 Behavior Requirements

982 6.5.2.7.3.1 Preconditions

983 `$instanceA` represents the instance of a `CIM_SoftwareInstallationService` and `$instanceB` represents
984 the instance of a `CIM_ComputerSystem`, both of which are referenced by `CIM_ServiceAffectsElement`.

```
985 $instanceA=<CIM_SoftwareInstallationService single instance>;
```

```
986 $instanceB=<CIM_ComputerSystem single instance>;
```

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

988 6.5.2.7.3.2 Pseudo Code

```
989 #propertylist[] = NULL;
990 if ( false == #all )
991 {
992     #propertylist[] = <array of mandatory non-key property names (see CIM
993     Requirements)>;
994 }
995 &smShowAssociationInstance ( "CIM_ServiceAffectsElement", $instanceA.getObjectPath(),
996     $instanceB.getObjectPath(), #propertylist[] );
997 &smEnd;
```

998 **6.6 CIM_ManagedElement**

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

1000 Table 6 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
 1001 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
 1002 target. Table 6 is for informational purposes only; in case of a conflict between Table 6 and requirements
 1003 detailed in the following sections, the text detailed in the following sections supersedes the information in
 1004 Table 6.

1005 **Table 6 – Command Verb Requirements for CIM_ManagedElement**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Shall	See 6.6.1.
reset	Not supported	
set	Not supported	
show	Not supported	
start	Not supported	
stop	Not supported	

1006 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `reset`,
 1007 `set`, `show`, `start`, and `stop`.

1008 **6.6.1 Load**

1009 This section describes how to implement the `load` verb when applied to an instance of
 1010 `CIM_ManagedElement`. Implementations shall support the use of the `load` verb with
 1011 `CIM_ManagedElement`.

1012 **6.6.1.1 Load Command Form to Install Software Using a URI**

1013 This section describes how to implement the `load` verb that is used to install or update software on an
 1014 instance of `CIM_ManagedElement`. The location of the software is represented by a URI. When the
 1015 `Install`, `Update`, `Repair`, `Reboot`, `Uninstall`, or `SilentMode` property is specified, the format is expected to
 1016 be in Boolean format as defined in [DSP0216](#). When the `Password`, `Log`, or `AdministrativeMode` is
 1017 specified, the format is expected to be in String format as defined in [DSP0216](#).

1018 **6.6.1.1.1 Command Form**

```

1019 load -source <URI>
1020     Install=<Install option value 1>
1021     Update=<Install option value 2>
1022     Repair=<Install option value 3>
1023     Reboot=<Install option value 4>
1024     Password=<Install option value 5>
1025     Uninstall=<Install option value 6>
1026     Log=<Install option value 7>
1027     SilentMode=<Install option value 8>
1028     AdministrativeMode=<Install option value 9> <CIM_ManagedElement single instance>
    
```

1029 **6.6.1.1.2 CIM Requirements**1030 `CIM_SoftwareInstallationService.InstallFromURI`1031 **6.6.1.1.3 Behavior Requirements**

```

1032 $MEinstance=<CIM_ManagedElement single instance>
1033 $uri = <URI>
1034 for #i < n
1035     {
1036         #InstallOptions[#i] = <Install option value #i>
1037     }
1038 #Error = &smOpAssociators(
1039     $instance.getObjectPath(),
1040     "CIM_ServiceAffectsElement",
1041     "CIM_SoftwareInstallationService",
1042     NULL,
1043     NULL,
1044     NULL,
1045     $SWInsSerInstancePaths[])
1046 if (0 != #Error.code)
1047     {
1048     &smProcessOpError (#Error);
1049     //includes &smEnd;
1050     }
1051 else if (SWInsSerInstancePaths.length() > 0)
1052     {
1053     for #i < n //n is the number of CIM_SoftwareInstallationService instances
1054     {
1055         $SWInsSerInstance = $SWInsSerInstancePaths[i];
1056         #Error= smInstallFromURI(InstallOptions[], uri, MEinstance->,
1057             SWInsSerInstance->);
1058     if (0 == #Error.code)
1059     {
1060         #i = n
1061         //goto &smEnd;
1062     }
1063     }
1064     }
1065 else
1066     {
1067     //unspecified return code, generic failure
1068     $OperationError = smNewInstance("CIM_Error");
1069     //CIM_ERR_FAILED
1070     $OperationError.CIMStatusCode = 1;
1071     //Other
1072     $OperationError.ErrorType = 1;
1073     //Low
1074     $OperationError.PerceivedSeverity = 2;
1075     $OperationError.OwningEntity = DMTF:SMCLP;
1076     $OperationError.MessageID = 0x00000001;
1077     $OperationError.Message = "Operation is not supported";
1078     &smAddError($job, $OperationError);
1079     &smMakeCommandStatus($job);
1080     }
1081 &smEnd;

```


1082 6.6.1.2 Load Command Form to Install Software Using a Software Identity

1083 This section describes how to implement the `load` verb that is used to install or update an instance of
1084 `CIM_SoftwareIdentity` on an instance of `CIM_ManagedElement`.

1085 When the `Install`, `Update`, `Repair`, `Reboot`, `Uninstall`, or `SilentMode` property is specified, the format is
1086 expected to be in Boolean format as defined in [DSP0216](#). When the `Password`, `Log`, or
1087 `AdministrativeMode` is specified, the format is expected to be in String format as defined in [DSP0216](#).

1088 6.6.1.2.1 Command Form

```
1089 load -source <CIM_SoftwareIdentity single instance>
1090     Install=<Install option value 1>
1091     Update=<Install option value 2>
1092     Repair=<Install option value 3>
1093     Reboot=<Install option value 4>
1094     Password=<Install option value 5>
1095     Uninstall=<Install option value 6>
1096     Log=<Install option value 7>
1097     SilentMode=<Install option value 8>
1098     AdministrativeMode=<Install option value 9>
1099     <CIM_ManagedElement single instance>
```

1100 6.6.1.2.2 CIM Requirements

```
1101 CIM_SoftwareInstallationService.InstallFromSoftwareIdentity
```

1102 6.6.1.2.3 Behavior Requirements

```
1103 $MEinstance=<CIM_ManagedElement single instance>
1104 $SWinstance=<CIM_SoftwareIdentity single instance>
1105 for #i < n
1106     {
1107         #InstallOptions[#i] = <Install option value #i>
1108     }
1109 #Error = &smOpAssociators (
1110     $instance.getObjectPath(),
1111     "CIM_ServiceAffectsElement",
1112     "CIM_SoftwareInstallationService",
1113     NULL,
1114     NULL,
1115     NULL,
1116     $SWInsSerInstancePaths[] )
1117 if ( 0 != #Error.code )
1118     {
1119         &smProcessOpError (#Error);
1120         //includes &smEnd;
1121     }
1122 else if ( SWInsSerInstancePaths.length() > 0 )
1123     {
1124         for #i < n //n is the number of CIM_SoftwareInstallationService instances
1125             {
1126                 $SWInsSerInstance = $SWInsSerInstancePaths[i];
1127                 #Error=smInstallFromSoftwareIdentity ( #InstallOptions[],
1128                 $SWinstance.getObjectPath(), $MEinstance.getObjectPath(),
1129                 $SWInsSerInstance.getObjectPath() );
1130             if ( 0 == #Error.code )
1131                 {
1132                     #i = n
```

```
1133     //goto &smEnd;
1134   }
1135   else
1136   {
1137     //unspecified return code, generic failure
1138     $OperationError = smNewInstance("CIM_Error");
1139     //CIM_ERR_FAILED
1140     $OperationError.CIMStatusCode = 1;
1141     //Other
1142     $OperationError.ErrorType = 1;
1143     //Low
1144     $OperationError.PerceivedSeverity = 2;
1145     $OperationError.OwningEntity = DMTF:SMCLP;
1146     $OperationError.MessageID = 0x00000001;
1147     $OperationError.Message = "Operation is not supported";
1148     &smAddError($job, $OperationError);
1149     &smMakeCommandStatus($job);
1150   }
1151 &smEnd;
```

**ANNEX A
(informative)**

Change Log

1152
1153
1154
1155
1156

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

1157