



1
2
3
4

Document Number: DSP0805

Date: 2009-07-14

Version: 1.0.0

5 **Sensors Profile to 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27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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	5.1 ISetNumericSensorThreshold	9
44	5.2 IRestoreDefaultThresholds	10
45	6 Mappings.....	13
46	6.1 CIM_Sensor	13
47	6.2 CIM_NumericSensor.....	18
48	6.3 CIM_EnabledLogicalElementCapabilities.....	25
49	6.4 CIM_ElementCapabilities	27
50	6.5 CIM_SystemDevice	29
51	6.6 CIM_AssociatedSensor	32
52	ANNEX A (informative) Change Log	35
53		

54 Tables

55	Table 1 – Command Verb Requirements for CIM_Sensor	13
56	Table 2 – Command Verb Requirements for CIM_NumericSensor.....	18
57	Table 3 – Command Verb Requirements for CIM_EnabledLogicalElementCapabilities.....	25
58	Table 4 – Command Verb Requirements for CIM_ElementCapabilities	27
59	Table 5 – Command Verb Requirements for CIM_SystemDevice	30
60	Table 6 – Command Verb Requirements for CIM_AssociatedSensor	32
61		

63

Foreword

64 The *Sensors Profile to SM CLP Mapping Specification* (DSP0805) was prepared by the Server
65 Management Working Group.

66 **Conventions**

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

69 **Acknowledgements**

70 The authors wish to acknowledge the following participants from the DTMF Server Management Working
71 Group:

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

79

80

Introduction

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

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

86

Sensors Profile to SM CLP Mapping Specification

87 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 [Sensors Profile](#).

90 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.

94 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

97 DMTF DSP1009, *Sensors Profile 1.0*,
98 http://www.dmtf.org/standards/published_documents/DSP1009_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

101 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?func=ll&objId=4230456&objAction=browse&sort=subtype>

104 3 Terms and Definitions

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

106 3.1

107 **can**

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

109 3.2

110 **cannot**

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

112 3.3

113 **conditional**

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

116 3.4

117 **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 This section details any acronyms or abbreviations used in this document where there is an expectation
146 that they may not be well-known to a reader.

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

- 159 **4.5**
 160 **SM**
 161 Server Management
- 162 **4.6**
 163 **SMI-S**
 164 Storage Management Initiative
- 165 **4.7**
 166 **SNIA**
 167 Storage Networking Industry Association
- 168 **4.8**
 169 **UFsT**
 170 User Friendly selection Tag

171 **5 Recipes**

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

- 174 • smResetRSC
- 175 • smShowInstance
- 176 • smShowInstances
- 177 • smShowAssociationInstance
- 178 • smShowAssociationInstances
- 179 • smStartRSC
- 180 • smStopRSC

181 **5.1 ISetNumericSensorThreshold**

182 **5.1.1 Description**

183 This method is used to modify the threshold properties of an instance of CIM_NumericSensor.

184 **5.1.2 Preconditions**

185 \$job contains the CIM_ConcreteJob instance for the operation.

186 **5.1.3 Pseudo Code**

```

187 sub void ISetNumericSensorThreshold ( string #thresholdName, string #requestedValue,
188     $target-> ) {
189     #validThresholdName = false;
190     for #i in $target->SettableThresholds[]
191     {
192         if ( #i == #thresholdName )
193         {
194             #validThresholdName = true;
195         }
196     }
  
```

```

197     if ( ! #validThresholdName )
198     {
199         $OperationError = smNewInstance("CIM_Error");
200         //CIM_ERR_FAILED
201         $OperationError.CIMStatusCode = 1;
202         //Software Error
203         $OperationError.ErrorType = 4;
204         //Low
205         $OperationError.PerceivedSeverity = 2;
206         $OperationError.OwningEntity = DMTF:SMCLP;
207         $OperationError.MessageID = 0x0000000E;
208         $OperationError.Message = "The target property name is invalid.";
209         &smAddError($job, $OperationError);
210         &smMakeCommandStatus($job);
211         &smEnd;
212     }
213     else
214     {
215         $target-><#thresholdName> = #requestedValue;
216         #propertyList[1] = #thresholdName;
217         #Error = &smOpModifyInstance ( $target, #propertyList );
218         if (0 != Error.code)
219         {
220             &smProcessOpError (#Error);
221             //includes end;
222         }
223     }
224     &smCommandCompleted( $job );
225     &smDisplayInstance ( $target );
226     &smEnd;
227 }

```

228 5.2 IRestoreDefaultThresholds

229 5.2.1 Description

230 This method is used to modify the threshold properties of an instance of CIM_NumericSensor.

231 5.2.2 Preconditions

232 \$job contains the CIM_ConcreteJob instance for the operation.

233 5.2.3 Pseudo-Code

```

234 sub void lRestoreDefaultThresholds ( $target-> )
235 {
236     %InArguments[] = {};
237     %OutArguments[] = {};
238     #Error = InvokeMethod ( $target->,
239                             "RestoreDefaultThresholds",
240                             %InArguments[],
241                             %OutArguments[],
242                             #returnStatus);

```

```

243     if (0 != #Error.code)
244     {
245         //method invocation failed
246         if ( (null != #Error.$error) && (null != #Error.$error[0]) )
247         {
248             //if the method invocation contains an embedded error
249             //use it for the Error for the overall job
250             &smAddError($job, #Error.$error[0]);
251             &smMakeCommandStatus($job);
252             &smEnd;
253         }
254         else if ( 17 == #Error.code ) {
255             //17 - CIM_ERR_METHOD_NOT_FOUND
256             // The specified extrinsic method does not exist.
257             $OperationError = smNewInstance("CIM_Error");
258             // CIM_ERR_METHOD_NOT_FOUND
259             $OperationError.CIMStatusCode = 17;
260             //Software Error
261             $OperationError.ErrorType = 10;
262             //Unknown
263             $OperationError.PerceivedSeverity = 0;
264             $OperationError.OwningEntity = DMTF:SMCLP;
265             $OperationError.MessageID = 0x00000001;
266             $OperationError.Message = "Operation is not supported."
267             &smAddError($job, $OperationError);
268             &smMakeCommandStatus($job);
269             &smEnd;
270         }
271         else
272         {
273             //operation failed, but no detailed error instance, need to make one up
274             //make an Error instance and associate with job for Operation
275             $OperationError = smNewInstance("CIM_Error");
276             //CIM_ERR_FAILED
277             $OperationError.CIMStatusCode = 1;
278             //Software Error
279             $OperationError.ErrorType = 4;
280             //Unknown
281             $OperationError.PerceivedSeverity = 0;
282             $OperationError.OwningEntity = DMTF:SMCLP;
283             $OperationError.MessageID = 0x00000009;
284             $OperationError.Message = "An internal software error has occurred.";
285             &smAddError($job, $OperationError);
286             &smMakeCommandStatus($job);
287             &smEnd;
288         }
289     } //if CIM op failed
290     else if (0 == #returnStatus) {
291         //completed successfully
292         &smCommandCompleted($job);
293         &smEnd;
294     }
295     else if (1 == #returnStatus) {

```

```
296 //unsupported
297 $OperationError = smNewInstance("CIM_Error");
298 //CIM_ERR_NOT_SUPPORTED
299 $OperationError.CIMStatusCode = 7;
300 //Other
301 $OperationError.ErrorType = 1;
302 //Low
303 $OperationError.PerceivedSeverity = 2;
304 $OperationError.OwningEntity = DMTF:SMCLP;
305 $OperationError.MessageID = 0x00000001;
306 $OperationError.Message = "Operation is not supported.";
307 &smAddError($job, $OperationError);
308 &smMakeCommandStatus($job);
309 &smEnd;
310 }
311 else if (2 == #returnStatus) {
312 //generic failure
313 $OperationError = smNewInstance("CIM_Error");
314 //CIM_ERR_FAILED
315 $OperationError.CIMStatusCode = 1;
316 //Other
317 $OperationError.ErrorType = 1;
318 //Low
319 $OperationError.PerceivedSeverity = 2;
320 $OperationError.OwningEntity = DMTF:SMCLP;
321 $OperationError.MessageID = 0x00000002;
322 $OperationError.Message = "Failed. No further information is available.";
323 &smAddError($job, $OperationError);
324 &smMakeCommandStatus($job);
325 }
326 else {
327 //unspecified return code, generic failure
328 $OperationError = smNewInstance("CIM_Error");
329 //CIM_ERR_FAILED
330 $OperationError.CIMStatusCode = 1;
331 //Other
332 $OperationError.ErrorType = 1;
333 //Low
334 $OperationError.PerceivedSeverity = 2;
335 $OperationError.OwningEntity = DMTF:SMCLP;
336 $OperationError.MessageID = 0x00000002;
337 $OperationError.Message = "Failed. No further information is available.";
338 &smAddError($job, $OperationError);
339 &smMakeCommandStatus($job);
340 &smEnd;
341 }
342 }
```

343 6 Mappings

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

347 6.1 CIM_Sensor

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

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

354 **Table 1 – Command Verb Requirements for CIM_Sensor**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	May	See 6.1.2.
Set	May	See 6.1.3.
Show	Shall	See 6.1.4.
Start	May	See 6.1.5.
Stop	May	See 6.1.6.

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

356 6.1.1 Ordering of Results

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

- 359 • Results for `CIM_Sensor` are unordered; therefore, no algorithm is defined.

360 6.1.2 Reset

361 This section describes how to implement the `reset` verb when applied to an instance of `CIM_Sensor`.
 362 Implementations may support the use of the `reset` verb with `CIM_Sensor`.

363 6.1.2.1 Command Form

364 `reset <CIM_Sensor single object>`

365 6.1.2.2 CIM Requirements

```

366 uint16 EnabledState;
367 uint16 RequestedState;
368 uint32 CIM_Sensor.RequestStateChange (
369     [IN] uint16 RequestedState,
370     [OUT] REF CIM_ConcreteJob Job,
371     [IN] datetime TimeoutPeriod );

```

372 6.1.2.3 Behavior Requirements

373 6.1.2.3.1 Preconditions

374 \$instance represents the targeted instance of CIM_Sensor.

```

375 $instance=<CIM_Sensor single object>;

```

376 6.1.2.3.2 Pseudo Code

```

377 &smResetRSC ( $instance.getObjectPath() );
378 &smEnd;

```

379 6.1.3 Set

380 This section describes how to implement the `set` verb when it is applied to an instance of CIM_Sensor.
 381 Implementations may support the use of the `set` verb with CIM_Sensor.

382 The `set` verb is used to modify descriptive properties of the CIM_Sensor instance.

383 6.1.3.1 General Usage of Set for a Single Property

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

386 The requirement for supporting modification of a property using this command form shall be equivalent to
 387 the requirement for supporting modification of the property using the ModifyInstance operation as defined
 388 in the [Sensors Profile](#).

389 6.1.3.1.1 Command Form

```

390 set <CIM_Sensor single instance> <propertyname>=<propertyvalue>

```

391 6.1.3.1.2 CIM Requirements

392 See CIM_Sensor in the "CIM Elements" section of the [Sensors Profile](#) for the list of modifiable properties.

393 6.1.3.1.3 Behavior Requirements

```

394 $instance=<CIM_Sensor single instance>
395 #propertyName[] = {<propertyname>};
396 #propertyValues[] = {<propertyvalue>};
397 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
398 &smEnd;

```

399 6.1.3.2 General Usage of Set for Multiple Properties

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

403 The requirement for supporting modification of a property using this command form shall be equivalent to
404 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
405 in the [Sensors Profile](#).

406 6.1.3.2.1 Command Form

```
407 set <CIM_Sensor single instance> <propertyname1>=<propertyvalue1>  
408     <propertynamen>=<propertyvaluen>
```

409 6.1.3.2.2 CIM Requirements

410 See `CIM_Sensor` in the “CIM Elements” section of the [Sensors Profile](#) for the list of mandatory properties.

411 6.1.3.2.3 Behavior Requirements

```
412 $instance=<CIM_Sensor single instance>  
413 #propertyNames[] = {<propertyname>};  
414 for #i < n  
415     {  
416         #propertyNames[#i] = <propertyname#i>  
417         #propertyValues[#i] = <propertyvalue#i>  
418     }  
419 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );  
420 &smEnd;
```

421 6.1.4 Show

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

424 6.1.4.1 Show Command Form for Multiple Objects Target

425 This command form is used to show many instances of `CIM_Sensor`.

426 6.1.4.1.1 Command Form

```
427 show <CIM_Sensor multiple objects>
```

428 6.1.4.1.2 CIM Requirements

429 See `CIM_Sensor` in the “CIM Elements” section of the [Sensors Profile](#) for the list of mandatory properties.

430 6.1.4.1.3 Behavior Requirements

431 6.1.4.1.3.1 Preconditions

432 `$containerInstance` represents the instance of `CIM_ComputerSystem` which represents the
433 container system and is associated to the targeted instances of `CIM_Sensor` through the
434 `CIM_SystemDevice` association.

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

436 **6.1.4.1.3.2 Pseudo Code**

```

437 #propertylist[] = NULL;
438 if ( false == #all)
439     {
440         #propertylist[] = <array of mandatory non-key property names (see CIM
441             Requirements)>;
442     }
443 &smShowInstances ( "CIM_Sensor", "CIM_SystemDevice",
444     $containerInstance.getObjectPath(), #propertylist[] );
445 &smEnd;

```

446 **6.1.4.2 Show Command Form for Single Object Target**

447 This command form is used to show a single instance of CIM_Sensor.

448 **6.1.4.2.1 Command Form**

```

449 show <CIM_Sensor single object>

```

450 **6.1.4.2.2 CIM Requirements**

451 See CIM_Sensor in the "CIM Elements" section of the [Sensors Profile](#) for the list of mandatory properties.

452 **6.1.4.2.3 Behavior Requirements**453 **6.1.4.2.3.1 Preconditions**

454 \$instance represents the targeted instance of CIM_Sensor.

```

455 $instance=<CIM_Sensor single object>;

```

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

457 **6.1.4.2.3.2 Pseudo Code**

```

458 #propertylist[] = NULL;
459 if ( false == #all)
460     {
461         #propertylist[] = <array of mandatory non-key property names (see CIM
462             Requirements)>;
463     }
464 &smShowInstance ( $instance, #propertylist[] );
465 &smEnd;

```

466 **6.1.5 Start**

467 This section describes how to implement the `start` verb when applied to an instance of CIM_Sensor.
468 Implementations may support the use of the `start` verb with CIM_Sensor.

469 **6.1.5.1 Command Form**

```

470 start <CIM_Sensor single object>

```


471 6.1.5.2 CIM Requirements

```
472 uint16 EnabledState;
473 uint16 RequestedState;
474 uint32 CIM_Sensor.RequestStateChange (
475     [IN] uint16 RequestedState,
476     [OUT] REF CIM_ConcreteJob Job,
477     [IN] datetime TimeoutPeriod );
```

478 6.1.5.3 Behavior Requirements

479 6.1.5.3.1.1 Preconditions

480 \$instance represents the targeted instance of CIM_Sensor.

```
481 $instance=<CIM_Sensor single object>;
```

482 6.1.5.3.1.2 Pseudo Code

```
483 &smStartRSC ( $instance.getObjectPath() );
484 &smEnd;
```

485 6.1.6 Stop

486 This section describes how to implement the `stop` verb when applied to an instance of CIM_Sensor.
487 Implementations may support the use of the `stop` verb with CIM_Sensor.

488 6.1.6.1 Command Form

```
489 stop <CIM_Sensor single object>
```

490 6.1.6.2 CIM Requirements

```
491 uint16 EnabledState;
492 uint16 RequestedState;
493 uint32 CIM_Sensor.RequestStateChange (
494     [IN] uint16 RequestedState,
495     [OUT] REF CIM_ConcreteJob Job,
496     [IN] datetime TimeoutPeriod );
```

497 6.1.6.3 Behavior Requirements

498 6.1.6.3.1 Preconditions

499 \$instance represents the targeted instance of CIM_Sensor.

```
500 $instance=<CIM_Sensor single object>;
```

501 6.1.6.3.2 Pseudo Code

```
502 &smStopRSC ( $instance.getObjectPath() );
503 &smEnd;
```

504 6.2 CIM_NumericSensor

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

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

511 **Table 2 – Command Verb Requirements for CIM_NumericSensor**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	May	See 6.2.1.
Set	May	See 6.2.2.
Show	Shall	See 6.2.3.
Start	May	See 6.2.4.
Stop	May	See 6.2.5.

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

513 6.2.1 Reset

514 This section describes how to implement the `reset` verb when applied to an instance of
 515 `CIM_NumericSensor`. Implementations may support the use of the `reset` verb with `CIM_NumericSensor`.

516 6.2.1.1 Command Form

```
517 reset <CIM_NumericSensor single object>
```

518 6.2.1.2 CIM Requirements

```
519 uint16 EnabledState;  

  520 uint16 RequestedState;  

  521 uint32 CIM_NumericSensor.RequestStateChange (  

  522     [IN] uint16 RequestedState,  

  523     [OUT] REF CIM_ConcreteJob Job,  

  524     [IN] datetime TimeoutPeriod );
```

525 6.2.1.3 Behavior Requirements

526 6.2.1.3.1 Preconditions

527 `$instance` represents the targeted instance of `CIM_NumericSensor`.

```
528 $instance=<CIM_NumericSensor single object>;
```

529 6.2.1.3.2 Pseudo Code

```
530 &smResetRSC ( $instance.getObjectPath() );
531 &smEnd;
```

532 6.2.2 Set

533 This section describes how to implement the `set` verb when it is applied to an instance of
534 `CIM_NumericSensor`. Implementations may support the use of the `set` verb with `CIM_NumericSensor`.

535 The `set` verb is used to modify descriptive properties of the `CIM_NumericSensor` instance.

536 6.2.2.1 General Usage of Set for a Single Property

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

539 The requirement for supporting modification of a property using this command form shall be equivalent to
540 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
541 in the [Sensors Profile](#).

542 6.2.2.1.1 Command Form

```
543 set <CIM_NumericSensor single instance> <propertyname>=<propertyvalue>
```

544 6.2.2.1.2 CIM Requirements

545 See `CIM_NumericSensor` in the “CIM Elements” section of the [Sensors Profile](#) for the list of modifiable
546 properties.

547 6.2.2.1.3 Behavior Requirements

```
548 $instance=<CIM_NumericSensor single instance>
549 #propertyName[] = {<propertyname>};
550 #propertyValues[] = {<propertyvalue>};
551 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
552 &smEnd;
```

553 6.2.2.2 General Usage of Set for Multiple Properties

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

557 The requirement for supporting modification of a property using this command form shall be equivalent to
558 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
559 in the [Sensors Profile](#).

560 6.2.2.2.1 Command Form

```
561 set <CIM_NumericSensor single instance> <propertyname1>=<propertyvalue1>
562 <propertynamen>=<propertyvaluen>
```

563 6.2.2.2.2 CIM Requirements

564 See `CIM_NumericSensor` in the “CIM Elements” section of the [Sensors Profile](#) for the list of mandatory
565 properties.

566 6.2.2.2.3 Behavior Requirements

```

567 $instance=<CIM_NumericSensor single instance>
568 #propertyNames[] = {<propertyname>};
569 for #i < n
570     {
571         #propertyNames[#i] = <propertyname#i>
572         #propertyValues[#i] = <propertyvalue#i>
573     }
574 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
575 &smEnd;
```

576 6.2.2.3 Set Command Form for the LowerThresholdNonCritical Property

577 This command form is used to modify the LowerThresholdNonCritical property of the instances of
 578 CIM_NumericSensor.

579 6.2.2.3.1 Command Form

```
580 set <CIM_NumericSensor single object> LowerThresholdNonCritical=<request value>
```

581 6.2.2.3.2 CIM Requirements

```
582 sint32 LowerThresholdNonCritical;
```

583 6.2.2.3.3 Behavior Requirements

584 6.2.2.3.3.1 Preconditions

585 \$instance represents the targeted instance of CIM_NumericSensor.

```
586 $instance=<CIM_NumericSensor single object>;
```

587 6.2.2.3.3.2 Pseudo Code

```

588 &lSetNumericSensorThreshold ( "LowerThresholdNonCritical", <request value>,
589     $instance.getObjectPath() );
590 &smEnd;
```

591 6.2.2.4 Set Command Form for the UpperThresholdNonCritical Property

592 This command form is used to modify the UpperThresholdNonCritical property of the instances of
 593 CIM_NumericSensor.

594 6.2.2.4.1 Command Form

```
595 set <CIM_NumericSensor single object> UpperThresholdNonCritical=<request value>
```

596 6.2.2.4.2 CIM Requirements

```
597 sint32 UpperThresholdNonCritical;
```

598 6.2.2.4.3 Behavior Requirements

599 6.2.2.4.3.1 Preconditions

600 \$instance represents the targeted instance of CIM_NumericSensor.

```
601 $instance=<CIM_NumericSensor single object>;
```

602 **6.2.2.4.3.2 Pseudo Code**

```
603 &lSetNumericSensorThreshold ( "UpperThresholdNonCritical", <request value>,
604     $instance.getObjectPath() );
605 &smEnd;
```

606 **6.2.2.5 Set Command Form for the LowerThresholdCritical Property**

607 This command form is used to modify the LowerThresholdCritical property of the instances of
608 CIM_NumericSensor.

609 **6.2.2.5.1 Command Form**

```
610 set <CIM_NumericSensor single object> LowerThresholdCritical=<request value>
```

611 **6.2.2.5.2 CIM Requirements**

```
612 sint32 LowerThresholdCritical;
```

613 **6.2.2.5.3 Behavior Requirements**614 **6.2.2.5.3.1 Preconditions**

615 \$instance represents the targeted instance of CIM_NumericSensor.

```
616 $instance=<CIM_NumericSensor single object>;
```

617 **6.2.2.5.3.2 Pseudo Code**

```
618 &lSetNumericSensorThreshold ( "LowerThresholdCritical", <request value>,
619     $instance.getObjectPath() );
620 &smEnd;
```

621 **6.2.2.6 Set Command Form for the UpperThresholdCritical Property**

622 This command form is used to modify the UpperThresholdCritical property of the instances of
623 CIM_NumericSensor.

624 **6.2.2.6.1 Command Form**

```
625 set <CIM_NumericSensor single object> UpperThresholdCritical=<request value>
```

626 **6.2.2.6.2 CIM Requirements**

```
627 sint32 UpperThresholdCritical;
```

628 **6.2.2.6.3 Behavior Requirements**629 **6.2.2.6.3.1 Preconditions**

630 \$instance represents the targeted instance of CIM_NumericSensor.

```
631 $instance=<CIM_NumericSensor single object>;
```

632 **6.2.2.6.3.2 Pseudo Code**

```
633 &lSetNumericSensorThreshold ( "UpperThresholdCritical", <request value>,
634     $instance.getObjectPath() );
635 &smEnd;
```

636 **6.2.2.7 Set Command Form for the LowerThresholdFatal Property**

637 This command form is used to modify the LowerThresholdFatal property of the instances of
638 CIM_NumericSensor.

639 **6.2.2.7.1 Command Form**

```
640 set <CIM_NumericSensor single object> LowerThresholdFatal=<request value>
```

641 **6.2.2.7.2 CIM Requirements**

```
642 sint32 LowerThresholdFatal;
```

643 **6.2.2.7.3 Behavior Requirements**

644 **6.2.2.7.3.1 Preconditions**

645 \$instance represents the targeted instance of CIM_NumericSensor.

```
646 $instance=<CIM_NumericSensor single object>;
```

647 **6.2.2.7.3.2 Pseudo Code**

```
648 &lSetNumericSensorThreshold ( "LowerThresholdFatal", <request value>,  
649     $instance.getObjectPath() );  
650 &smEnd;
```

651 **6.2.2.8 Set Command Form for UpperThresholdFatal Property**

652 This command form is used to modify UpperThresholdFatal property of the instances of
653 CIM_NumericSensor.

654 **6.2.2.8.1 Command Form**

```
655 set <CIM_NumericSensor single object> UpperThresholdFatal=<request value>
```

656 **6.2.2.8.2 CIM Requirements**

```
657 sint32 UpperThresholdFatal;
```

658 **6.2.2.8.3 Behavior Requirements**

659 **6.2.2.8.3.1 Preconditions**

660 \$instance represents the targeted instance of CIM_NumericSensor.

```
661 $instance=<CIM_NumericSensor single object>;
```

662 **6.2.2.8.3.2 Pseudo Code**

```
663 &lSetNumericSensorThreshold ( "UpperThresholdFatal", <request value>,  
664     $instance.getObjectPath() );  
665 &smEnd;
```

666 **6.2.2.9 Set Command Form for Restoring Threshold Defaults**

667 This command form is used to restore the values of threshold property of the instance of
668 CIM_NumericSensor to the defaults.

669 **6.2.2.9.1 Command Form**

```
670 set -default <CIM_NumericSensor single object> SettableThresholds
```

671 **6.2.2.9.2 CIM Requirements**

```
672 uint32 CIM_Sensor.RestoreDefaultThresholds();
```

673 **6.2.2.9.3 Behavior Requirements**674 **6.2.2.9.3.1 Preconditions**

675 \$instance represents the targeted instance of CIM_NumericSensor.

```
676 $instance=<CIM_NumericSensor single object>;
```

677 **6.2.2.9.3.2 Pseudo Code**

```
678 &lRestoreDefaultThresholds ( $instance.getObjectPath() );
679 &smEnd;
```

680 **6.2.3 Show**

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

683 **6.2.3.1 Show Command Form for Multiple Objects Target**

684 This command form is used to show many instances of CIM_NumericSensor.

685 **6.2.3.1.1 Command Form**

```
686 show <CIM_NumericSensor multiple objects>
```

687 **6.2.3.1.2 CIM Requirements**

688 See CIM_NumericSensor in the “CIM Elements” section of the [Sensors Profile](#) for the list of mandatory
689 properties.

690 **6.2.3.1.3 Behavior Requirements**691 **6.2.3.1.3.1 Preconditions**

692 \$containerInstance represents the instance of CIM_ComputerSystem which represents the
693 container system and is associated to the targeted instances of CIM_NumericSensor through the
694 CIM_SystemDevice association.

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

696 **6.2.3.1.3.2 Pseudo Code**

```
697 #propertylist[] = NULL;
698 if ( false == #all)
699     {
700         #propertylist[] = <array of mandatory non-key property names (see CIM
701             Requirements)>;
702     }
703 &smShowInstances ( "CIM_NumericSensor", "CIM_ElementCapabilities",
704     $containerInstance.getObjectPath(), #propertylist[] );
705 &smEnd;
```

706 6.2.3.2 Show Command Form for a Single Object Target

707 This command form is used to show a single instance of CIM_NumericSensor.

708 6.2.3.2.1 Command Form

```
709 show <CIM_NumericSensor single object>
```

710 6.2.3.2.2 CIM Requirements

711 See CIM_NumericSensor in the “CIM Elements” section of the [Sensors Profile](#) for the list of mandatory
712 properties.

713 6.2.3.2.3 Behavior Requirements

714 6.2.3.2.3.1 Preconditions

715 \$instance represents the targeted instance of CIM_NumericSensor.

```
716 $instance=<CIM_NumericSensor single object>;
```

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

718 6.2.3.2.3.2 Pseudo Code

```
719 #propertylist[] = NULL;
720 if ( false == #all)
721     {
722         #propertylist[] = <array of mandatory non-key property names (see CIM
723             Requirements)>;
724     }
725 &smShowInstance ( $instance, #propertylist[] );
726 &smEnd;
```

727 6.2.4 Start

728 This section describes how to implement the `start` verb when applied to an instance of
729 CIM_NumericSensor. Implementations may support the use of the `start` verb with CIM_NumericSensor.

730 6.2.4.1 Command Form

```
731 start <CIM_NumericSensor single object>
```

732 6.2.4.2 CIM Requirements

```
733 uint16 EnabledState;
734 uint16 RequestedState;
735 uint32 CIM_NumericSensor.RequestStateChange (
736     [IN] uint16 RequestedState,
737     [OUT] REF CIM_ConcreteJob Job,
738     [IN] datetime TimeoutPeriod );
```

739 6.2.4.3 Behavior Requirements

740 6.2.4.3.1 Preconditions

741 \$instance represents the targeted instance of CIM_NumericSensor.

```
742 $instance=<CIM_NumericSensor single object>;
```


743 **6.2.4.3.2 Pseudo Code**

```
744 &smStartRSC ( $instance.GetObjectPath() );
745 &smEnd;
```

746 **6.2.5 Stop**

747 This section describes how to implement the `stop` verb when applied to an instance of
 748 `CIM_NumericSensor`. Implementations may support the use of the `stop` verb with `CIM_NumericSensor`.

749 **6.2.5.1 Command Form**

```
750 stop <CIM_NumericSensor single object>
```

751 **6.2.5.2 CIM Requirements**

```
752 uint16 EnabledState;
753 uint16 RequestedState;
754 uint32 CIM_NumericSensor.RequestStateChange (
755     [IN] uint16 RequestedState,
756     [OUT] REF CIM_ConcreteJob Job,
757     [IN] datetime TimeoutPeriod );
```

758 **6.2.5.3 Behavior Requirements**

759 **6.2.5.3.1 Preconditions**

760 In this section `$instance` represents the targeted instance of `CIM_NumericSensor`.

```
761 $instance=<CIM_NumericSensor single object>;
```

762 **6.2.5.3.2 Pseudo Code**

```
763 &smStopRSC ( $instance.GetObjectPath() );
764 &smEnd;
```

765 **6.3 CIM_EnabledLogicalElementCapabilities**

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

767 Table 3 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
 768 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
 769 target. Table 3 is for informational purposes only: in case of a conflict between Table 3 and requirements
 770 detailed in the following sections, the text detailed in the following sections supersedes the information in
 771 Table 3.

772 **Table 3 – 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	

Command Verb	Requirement	Comments
Set	Not supported	
Show	Shall	See 6.3.1.
Start	Not supported	
Stop	Not supported	

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

775 6.3.1 Show

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

779 6.3.1.1 Show Command Form for Multiple Objects Target

780 This command form is used to show many instances of `CIM_EnabledLogicalElementCapabilities`.

781 6.3.1.1.1 Command Form

```
782 show <CIM_EnabledLogicalElementCapabilities multiple objects>
```

783 6.3.1.1.2 CIM Requirements

784 See `CIM_EnabledLogicalElementCapabilities` in the “CIM Elements” section of the [Sensors Profile](#) for the
785 list of mandatory properties.

786 6.3.1.1.3 Behavior Requirements

787 6.3.1.1.3.1 Preconditions

788 `$containerInstance` represents the instance of `CIM_ConcreteCollection` with `ElementName` property
789 that contains “Capabilities” and is associated to the targeted instances of
790 `CIM_EnabledLogicalElementCapabilities` through the `CIM_MemberOfCollection` association.

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

792 6.3.1.1.3.2 Pseudo Code

```
793 #propertylist[] = NULL;
794 if ( false == #all)
795 {
796     #propertylist[] = <array of mandatory non-key property names (see CIM
797         Requirements)>;
798 }
799 &smShowInstances ( "CIM_EnabledLogicalElementCapabilities", "CIM_MemberOfCollection",
800     $containerInstance.getObjectPath(), #propertylist[] );
801 &smEnd;
```

802 6.3.1.2 Show Command Form for Single Object Target

803 This command form is used to show a single instance of `CIM_EnabledLogicalElementCapabilities`.

804 **6.3.1.2.1 Command Form**

805 `show <CIM_EnabledLogicalElementCapabilities single object>`

806 **6.3.1.2.2 CIM Requirements**

807 See CIM_EnabledLogicalElementCapabilities in the “CIM Elements” section of the [Sensors Profile](#) for the
808 list of mandatory properties.

809 **6.3.1.2.3 Behavior Requirements**

810 **6.3.1.2.3.1 Preconditions**

811 \$instance represents the targeted instance of CIM_EnabledLogicalElementCapabilities.

812 `$instance=<CIM_EnabledLogicalElementCapabilities single object>;`

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

814 **6.3.1.2.3.2 Pseudo Code**

```
815 #propertylist[] = NULL;
816 if ( false == #all) {
817     #propertylist[] = <array of mandatory non-key property names (see CIM
818         Requirements)>;
819 }
820 &smShowInstance ( $instance, #propertylist[] );
821 &smEnd;
```

822 **6.4 CIM_ElementCapabilities**

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

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

829 **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.
Start	Not supported	
Stop	Not supported	

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

832 6.4.1 Ordering of Results

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

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

836 6.4.2 Show

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

840 6.4.2.1 Show Command Form for Multiple Objects Target – 841 CIM_EnabledLogicalElementCapabilities Reference

842 This command form is used to show many instances of CIM_ElementCapabilities. This command form
843 corresponds to a `show` command issued against instances of CIM_ElementCapabilities where only one
844 reference is specified and the reference is to the scoping instance of
845 CIM_EnabledLogicalElementCapabilities.

846 6.4.2.1.1 Command Form

```
847 show <CIM_ElementCapabilities multiple objects>
```

848 6.4.2.1.2 CIM Requirements

849 See CIM_ElementCapabilities in the “CIM Elements” section of the [Sensors Profile](#) for the list of
850 mandatory properties.

851 6.4.2.1.3 Behavior Requirements

852 6.4.2.1.3.1 Preconditions

853 `$instance` represents the instance of CIM_EnabledLogicalElementCapabilities which is referenced by
854 CIM_ElementCapabilities.

855 6.4.2.1.3.2 Pseudo Code

```
856 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
857 &smEnd;
```

858 6.4.2.2 Show Command Form for Multiple Objects – CIM_Sensor or CIM_NumericSensor 859 Reference

860 This command form is used to show a single instance of CIM_ElementCapabilities. This command form
861 corresponds to a `show` command issued against a single instance of CIM_ElementCapabilities where
862 only one reference is specified and the reference is to the instance of CIM_Sensor or
863 CIM_NumericSensor.

864 6.4.2.2.1 Command Form

```
865 show <CIM_ElementCapabilities multiple objects>
```

866 6.4.2.2.2 CIM Requirements

867 See CIM_ElementCapabilities in the “CIM Elements” section of the [Sensors Profile](#) for the list of
868 mandatory properties.

869 6.4.2.2.3 Behavior Requirements

870 6.4.2.2.3.1 Preconditions

871 `$instance` represents the instance of `CIM_Sensor` or `CIM_NumericSensor` which is referenced by
872 `CIM_ElementCapabilities`.

873 6.4.2.2.3.2 Pseudo Code

```
874 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath(),
875     #propertylist[] );
876 &smEnd;
```

877 6.4.2.3 Show Command Form for a Single Object Target – Both References

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

881 6.4.2.3.1 Command Form

```
882 show <CIM_ElementCapabilities single object>
```

883 6.4.2.3.2 CIM Requirements

884 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [Sensors Profile](#) for the list of
885 mandatory properties.

886 6.4.2.3.3 Behavior Requirements

887 6.4.2.3.3.1 Preconditions

888 `$instanceA` represents the referenced instance of `CIM_Sensor` or `CIM_NumericSensor` through
889 `CIM_ElementCapabilities` association.

890 `$instanceB` represents the instance of `CIM_EnabledLogicalElementCapabilities` which is referenced by
891 `CIM_ElementCapabilities`.

892 6.4.2.3.3.2 Pseudo Code

```
893 &smShowAssociationInstance ( "CIM_ElementCapabilities", $instanceA.getObjectPath(),
894     $instanceB.getObjectPath() );
895 &smEnd;
```

896 6.5 CIM_SystemDevice

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

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

903

Table 5 – Command Verb Requirements for CIM_SystemDevice

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	

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

906 6.5.1 Ordering of Results

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

909 Results for *CIM_SystemDevice* are unordered; therefore, no algorithm is defined.

910 6.5.2 Show

911 This section describes how to implement the *show* verb when applied to an instance of
 912 *CIM_SystemDevice*. Implementations shall support the use of the *show* verb with *CIM_SystemDevice*.

913 6.5.2.1 Show Command Form for Multiple Objects Target – *CIM_ComputerSystem* Reference

914 This command form is used to show many instances of *CIM_SystemDevice*. This command form
 915 corresponds to a *show* command issued against the instance of *CIM_SystemDevice* where only one
 916 reference is specified and the reference is to the scoping instance of *CIM_ComputerSystem*.

917 6.5.2.1.1 Command Form

918 `show <CIM_SystemDevice multiple objects>`

919 6.5.2.1.2 CIM Requirements

920 See *CIM_SystemDevice* in the “CIM Elements” section of the [Sensors Profile](#) for the list of mandatory
 921 properties.

922 6.5.2.1.3 Behavior Requirements

923 6.5.2.1.3.1 Preconditions

924 *§instance* represents the instance of a *CIM_ComputerSystem*, which is referenced by
 925 *CIM_SystemDevice*.

926 6.5.2.1.3.2 Pseudo Code

```
927 &smShowAssociationInstances ( "CIM_SystemDevice", $instance.getObjectPath());
928 &smEnd;
```

929 6.5.2.2 Show Command Form for Multiple Objects Target – CIM_Sensor or CIM_NumericSensor 930 Reference

931 This command form is used to show a single instance of CIM_SystemDevice. This command form
932 corresponds to a `show` command issued against a single instance of CIM_SystemDevice, where only one
933 reference is specified and the reference is to the instance of CIM_Sensor or CIM_NumericSensor.

934 6.5.2.2.1 Command Form

```
935 show <CIM_SystemDevice multiple objects>
```

936 6.5.2.2.2 CIM Requirements

937 See CIM_SystemDevice in the “CIM Elements” section of the [Sensors Profile](#) for the list of mandatory
938 properties.

939 6.5.2.2.3 Behavior Requirements

940 6.5.2.2.3.1 Preconditions

941 `$instance` represents the instance of CIM_Sensor or CIM_NumericSensor which is referenced by
942 CIM_SystemDevice.

943 6.5.2.2.3.2 Pseudo Code

```
944 &smShowAssociationInstances ( "CIM_SystemDevice", $instance.getObjectPath() );
945 &smEnd;
```

946 6.5.2.3 Show Command Form for a Single Object Target – Both References

947 This command form is for the `show` verb applied to a single instance. This command form corresponds to
948 a `show` command issued against CIM_SystemDevice where both references are specified and therefore
949 the desired instance is unambiguously identified.

950 6.5.2.3.1 Command Form

```
951 show <CIM_SystemDevice single object>
```

952 6.5.2.3.2 CIM Requirements

953 See CIM_SystemDevice in the “CIM Elements” section of the [Sensors Profile](#) for the list of mandatory
954 properties.

955 6.5.2.3.3 Behavior Requirements

956 6.5.2.3.3.1 Preconditions

957 `$instanceA` represents the referenced instance of CIM_Sensor or CIM_NumericSensor through
958 CIM_SystemDevice association.

959 `$instanceB` represents the instance of CIM_Sensor or CIM_NumericSensor which is referenced by
960 CIM_SystemDevice.

961 **6.5.2.3.3.2 Pseudo Code**

```

962 &smShowAssociationInstance ( "CIM_SystemDevice", $instanceA.getObjectPath(),
963     $instanceB.getObjectPath() );
964 &smEnd;

```

965 **6.6 CIM_AssociatedSensor**

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

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

972 **Table 6 – Command Verb Requirements for CIM_AssociatedSensor**

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	

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

975 **6.6.1 Ordering of Results**

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

978 Results for `CIM_AssociatedSensor` are unordered; therefore, no algorithm is defined.

979 **6.6.2 Show**

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

983 **6.6.2.1 Show Command Form for Multiple Objects Target – CIM_ManagedSystemElement
 984 Subclass Reference**

985 This command form is used to show many instances of `CIM_AssociatedSensor`. This command form
 986 corresponds to a `show` command issued against instances of `CIM_AssociatedSensor` where only one
 987 reference is specified and the reference is to the scoping instance of a subclass of
 988 `CIM_ManagedSystemElement`.

989 6.6.2.1.1 Command Form

```
990 show <CIM_AssociatedSensor multiple objects>
```

991 6.6.2.1.2 CIM Requirements

992 See CIM_AssociatedSensor in the “CIM Elements” section of the [Sensors Profile](#) for the list of mandatory
993 properties.

994 6.6.2.1.3 Behavior Requirements

995 6.6.2.1.3.1 Preconditions

996 \$instance represents the instance of a subclass of CIM_ManagedSystemElement which is referenced
997 by CIM_AssociatedSensor.

998 6.6.2.1.3.2 Pseudo Code

```
999 &smShowAssociationInstances ( "CIM_AssociatedSensor", $instance.getObjectPath() );  
1000 &smEnd;
```

1001 6.6.2.2 Show Command Form for Multiple Objects Target – CIM_Sensor or CIM_NumericSensor 1002 Reference

1003 This command form is used to show a single instance of CIM_AssociatedSensor. This command form
1004 corresponds to a show command issued against a single instance of CIM_AssociatedSensor where only
1005 one reference is specified and the reference is to the instance of CIM_Sensor or CIM_NumericSensor.

1006 6.6.2.2.1 Command Form

```
1007 show <CIM_AssociatedSensor multiple objects>
```

1008 6.6.2.2.2 CIM Requirements

1009 See CIM_AssociatedSensor in the “CIM Elements” section of the [Sensors Profile](#) for the list of mandatory
1010 properties.

1011 6.6.2.2.3 Behavior Requirements

1012 6.6.2.2.3.1 Preconditions

1013 \$instance represents the instance of CIM_Sensor or CIM_NumericSensor which is referenced by
1014 CIM_AssociatedSensor.

1015 6.6.2.2.3.2 Pseudo Code

```
1016 &smShowAssociationInstances ( "CIM_AssociatedSensor", $instance.getObjectPath() );  
1017 &smEnd;
```

1018 6.6.2.3 Show Command Form for a Single Object Target – Both References

1019 This command form is for the show verb applied to a single instance. This command form corresponds to
1020 a show command issued against CIM_AssociatedSensor where both references are specified and
1021 therefore the desired instance is unambiguously identified.

1022 6.6.2.3.1 Command Form

```
1023 show <CIM_AssociatedSensor single object>
```

1024 6.6.2.3.2 CIM Requirements

1025 See CIM_AssociatedSensor in the “CIM Elements” section of the [Sensors Profile](#) for the list of mandatory
1026 properties.

1027 6.6.2.3.3 Behavior Requirements**1028 6.6.2.3.3.1 Preconditions**

1029 \$instanceA represents the referenced instance of CIM_Sensor or CIM_NumericSensor through
1030 CIM_AssociatedSensor association.

1031 \$instanceB represents the instance of subclass of CIM_ManagedSystemElement which is referenced
1032 by CIM_AssociatedSensor.

1033 6.6.2.3.3.2 Pseudo Code

```
1034 &smShowAssociationInstance ( "CIM_AssociatedSensor", $instanceA.getObjectPath(),  
1035     $instanceB.getObjectPath() );  
1036 &smEnd;
```

1037

ANNEX A
(informative)

Change Log

1038
1039
1040
1041
1042

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

1043