Document Number: DSP0805

Date: 2009-07-14

Version: 1.0.0

Sensors Profile to SM CLP Mapping Specification

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

CONTENTS

Foreword 5
Introduction 6
1 Scope 7
2 Normative References 7
2.1 Approved References 7
2.2 Other References 7
3 Terms and Definitions 7
4 Symbols and Abbreviated Terms 8
5 Recipes 9
5.1 lSetNumericSensorThreshold 9
5.2 lRestoreDefaultThresholds 10
6 Mappings 13
6.1 CIM_Sensor 13
6.2 CIM_NumericSensor 18
6.3 CIM_EnabledLogicalElementCapabilities 25
6.4 CIM_ElementCapabilities 27
6.5 CIM_SystemDevice 29
6.6 CIM_AssociatedSensor 32
ANNEX A (informative) Change Log 35

Tables

Table 1 – Command Verb Requirements for CIM_Sensor 13
Table 2 – Command Verb Requirements for CIM_NumericSensor 18
Table 3 – Command Verb Requirements for CIM_EnabledLogicalElementCapabilities 25
Table 4 – Command Verb Requirements for CIM_ElementCapabilities 27
Table 5 – Command Verb Requirements for CIM_SystemDevice 30
Table 6 – Command Verb Requirements for CIM_AssociatedSensor 32

62

Foreword

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

Conventions

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

Acknowledgements

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

- Khachatur Papanyan – Dell Inc.
- Jon Hass – Dell Inc.
- Jeff Hilland – HP
- Christina Shaw – HP
- Aaron Merkin – IBM
- Perry Vincent – Intel
- John Leung – Intel

Introduction

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

The target audience for this specification is implementers of the SM CLP support for the *Sensors Profile*.

Sensors Profile to SM CLP Mapping Specification

1 Scope

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

2 Normative References

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

2.1 Approved References

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

DMTF DSP1009, *Sensors Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1009_1.0.pdf

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

2.2 Other References

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

3 Terms and Definitions

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

3.1
can
used for statements of possibility and capability, whether material, physical, or causal

3.2
cannot
used for statements of possibility and capability, whether material, physical or causal

3.3
conditional
indicates requirements to be followed strictly in order to conform to the document when the specified conditions are met

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

3.5
may
indicates a course of action permissible within the limits of the document

3.6
need not
indicates a course of action permissible within the limits of the document

3.7
optional
indicates a course of action permissible within the limits of the document

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

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

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

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

4 Symbols and Abbreviated Terms

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

4.1
CIM
Common Information Model

4.2
CLP
Command Line Protocol

4.3
DMTF
Distributed Management Task Force

4.4
IETF
Internet Engineering Task Force

4.5
SM
Server Management

4.6
SMI-S
Storage Management Initiative

4.7
SNIA
Storage Networking Industry Association

4.8
UFsT
User Friendly selection Tag

5 Recipes

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

- smResetRSC
- smShowInstance
- smShowInstances
- smShowAssociationInstance
- smShowAssociationInstances
- smStartRSC
- smStopRSC

5.1 lSetNumericSensorThreshold

5.1.1 Description

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

5.1.2 Preconditions

`$job` contains the CIM_ConcreteJob instance for the operation.

5.1.3 Pseudo Code

```
sub void lSetNumericSensorThreshold ( string #thresholdName, string #requestedValue,
      $target-> )  {
   #validThresholdName = false;
   for #i in $target->SettableThresholds[]
   {
      if ( #i == #thresholdName )
      {
         #validThresholdName = true;
      }
   }
```

```
   if ( ! #validThresholdName )
   {
      $OperationError = smNewInstance(“CIM_Error”);
      //CIM_ERR_FAILED
      $OperationError.CIMStatusCode = 1;
      //Software Error
      $OperationError.ErrorType = 4;
      //Low
      $OperationError.PerceivedSeverity = 2;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x0000000E;
      $OperationError.Message = “The target property name is invalid.”;
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else
   {
      $target-><#thresholdName> = #requestedValue;
      #propertyList[1] = #thresholdName;
      #Error = &smOpModifyInstance ( $target, #propertyList );
      if (0 != Error.code)
      {
         &smProcessOpError (#Error);
         //includes end;
      }
   }
   &smCommandCompleted( $job );
   &smDisplayInstance ( $target );
   &smEnd;
}
```

5.2 lRestoreDefaultThresholds

5.2.1 Description

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

5.2.2 Preconditions

`$job` contains the CIM_ConcreteJob instance for the operation.

5.2.3 Pseudo-Code

```
sub void lRestoreDefaultThresholds ( $target-> )
{
      %InArguments[] = {};
      %OutArguments[] = {};
      #Error = InvokeMethod ($target->,
                        “RestoreDefaultThresholds”,
                        %InArguments[],
                        %OutArguments[],
                        #returnStatus);
```

```
      if (0 != #Error.code)
      {
         //method invocation failed
         if ( (null != #Error.$error) && (null != #Error.$error[0]) )
         {
            //if the method invocation contains an embedded error
            //use it for the Error for the overall job
            &smAddError($job, #Error.$error[0]);
            &smMakeCommandStatus($job);
            &smEnd;
         }
         else if ( 17 == #Error.code ) {
            //17 – CIM_ERR_METHOD_NOT_FOUND
            // The specified extrinsic method does not exist.
            $OperationError = smNewInstance(“CIM_Error”);
            // CIM_ERR_METHOD_NOT_FOUND
            $OperationError.CIMStatusCode = 17;
            //Software Error
            $OperationError.ErrorType = 10;
            //Unknown
            $OperationError.PerceivedSeverity = 0;
            $OperationError.OwningEntity = DMTF:SMCLP;
            $OperationError.MessageID = 0x00000001;
            $OperationError.Message = “Operation is not supported.”
            &smAddError($job, $OperationError);
            &smMakeCommandStatus($job);
            &smEnd;
         }
         else
         {
            //operation failed, but no detailed error instance, need to make one up
            //make an Error instance and associate with job for Operation
            $OperationError = smNewInstance(“CIM_Error”);
            //CIM_ERR_FAILED
            $OperationError.CIMStatusCode = 1;
            //Software Error
            $OperationError.ErrorType = 4;
            //Unknown
            $OperationError.PerceivedSeverity = 0;
            $OperationError.OwningEntity = DMTF:SMCLP;
            $OperationError.MessageID = 0x00000009;
            $OperationError.Message = “An internal software error has occurred.”;
            &smAddError($job, $OperationError);
            &smMakeCommandStatus($job);
            &smEnd;
         }
      }//if CIM op failed
      else if (0 == #returnStatus)   {
         //completed successfully
         &smCommandCompleted($job);
         &smEnd;
      }
      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 }
```

6 Mappings

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

6.1 CIM_Sensor

The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in DSP0216.

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

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.

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

6.1.1 Ordering of Results

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

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

6.1.2 Reset

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

6.1.2.1 Command Form

```
reset <CIM_Sensor single object>
```

6.1.2.2 CIM Requirements

```
uint16 EnabledState;
uint16 RequestedState;
uint32 CIM_Sensor.RequestStateChange (
       [IN] uint16 RequestedState,
       [OUT] REF CIM_ConcreteJob Job,
       [IN] datetime TimeoutPeriod );
```

6.1.2.3 Behavior Requirements

6.1.2.3.1 Preconditions

`$instance` represents the targeted instance of CIM_Sensor.

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

6.1.2.3.2 Pseudo Code

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

6.1.3 Set

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

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

6.1.3.1 General Usage of Set for a Single Property

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

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

6.1.3.1.1 Command Form

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

6.1.3.1.2 CIM Requirements

See CIM_Sensor in the “CIM Elements” section of the *Sensors Profile* for the list of modifiable properties.

6.1.3.1.3 Behavior Requirements

```
$instance=<CIM_Sensor single instance>
#propertyNames[] = {<propertyname>};
#propertyValues[] = {<propertyvalue>};
&smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
&smEnd;
```

6.1.3.2 General Usage of Set for Multiple Properties

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

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

6.1.3.2.1 Command Form

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

6.1.3.2.2 CIM Requirements

See CIM_Sensor in the “CIM Elements” section of the *Sensors Profile* for the list of mandatory properties.

6.1.3.2.3 Behavior Requirements

```
$instance=<CIM_Sensor single instance>
#propertyNames[] = {<propertyname>};
for #i < n
   {
   #propertyNames[#i] = <propertname#i>
   #propertyValues[#i] = <propertyvalue#i>
   }
&smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
&smEnd;
```

6.1.4 Show

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

6.1.4.1 Show Command Form for Multiple Objects Target

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

6.1.4.1.1 Command Form

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

6.1.4.1.2 CIM Requirements

See CIM_Sensor in the “CIM Elements” section of the *Sensors Profile* for the list of mandatory properties.

6.1.4.1.3 Behavior Requirements

6.1.4.1.3.1 Preconditions

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

`#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>`**

6.1.5.2 CIM Requirements

```
uint16 EnabledState;
uint16 RequestedState;
uint32 CIM_Sensor.RequestStateChange (
       [IN] uint16 RequestedState,
       [OUT] REF CIM_ConcreteJob Job,
       [IN] datetime TimeoutPeriod );
```

6.1.5.3 Behavior Requirements

6.1.5.3.1.1 Preconditions

`$instance` represents the targeted instance of CIM_Sensor.

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

6.1.5.3.1.2 Pseudo Code

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

6.1.6 Stop

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

6.1.6.1 Command Form

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

6.1.6.2 CIM Requirements

```
uint16 EnabledState;
uint16 RequestedState;
uint32 CIM_Sensor.RequestStateChange (
       [IN] uint16 RequestedState,
       [OUT] REF CIM_ConcreteJob Job,
       [IN] datetime TimeoutPeriod );
```

6.1.6.3 Behavior Requirements

6.1.6.3.1 Preconditions

`$instance` represents the targeted instance of CIM_Sensor.

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

6.1.6.3.2 Pseudo Code

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

6.2 CIM_NumericSensor

The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in DSP0216.

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

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.

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

6.2.1 Reset

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

6.2.1.1 Command Form

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

6.2.1.2 CIM Requirements

```
uint16 EnabledState;
uint16 RequestedState;
uint32 CIM_NumericSensor.RequestStateChange (
       [IN] uint16 RequestedState,
       [OUT] REF CIM_ConcreteJob Job,
       [IN] datetime TimeoutPeriod );
```

6.2.1.3 Behavior Requirements

6.2.1.3.1 Preconditions

`$instance` represents the targeted instance of CIM_NumericSensor.

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

6.2.1.3.2 Pseudo Code

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

6.2.2 Set

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

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

6.2.2.1 General Usage of Set for a Single Property

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

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

6.2.2.1.1 Command Form

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

6.2.2.1.2 CIM Requirements

See CIM_NumericSensor in the “CIM Elements” section of the *Sensors Profile* for the list of modifiable properties.

6.2.2.1.3 Behavior Requirements

```
$instance=<CIM_NumericSensor single instance>
#propertyNames[] = {<propertyname>};
#propertyValues[] = {<propertyvalue>};
&smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
&smEnd;
```

6.2.2.2 General Usage of Set for Multiple Properties

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

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

6.2.2.2.1 Command Form

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

6.2.2.2.2 CIM Requirements

See CIM_NumericSensor in the “CIM Elements” section of the *Sensors Profile* for the list of mandatory 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] = <propertname#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>;
```

6.2.2.4.3.2 Pseudo Code

```
&lSetNumericSensorThreshold ( “UpperThresholdNonCritical”, <request value>,
      $instance.getObjectPath() );
&smEnd;
```

6.2.2.5 Set Command Form for the LowerThresholdCritical Property

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

6.2.2.5.1 Command Form

set <CIM_NumericSensor *single object*> LowerThresholdCritical=<*request value*>

6.2.2.5.2 CIM Requirements

```
sint32 LowerThresholdCritical;
```

6.2.2.5.3 Behavior Requirements

6.2.2.5.3.1 Preconditions

$instance represents the targeted instance of CIM_NumericSensor.

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

6.2.2.5.3.2 Pseudo Code

```
&lSetNumericSensorThreshold ( “LowerThresholdCritical”, <request value>,
      $instance.getObjectPath() );
&smEnd;
```

6.2.2.6 Set Command Form for the UpperThresholdCritical Property

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

6.2.2.6.1 Command Form

set <CIM_NumericSensor *single object*> UpperThresholdCritical=<*request value*>

6.2.2.6.2 CIM Requirements

```
sint32 UpperThresholdCritical;
```

6.2.2.6.3 Behavior Requirements

6.2.2.6.3.1 Preconditions

$instance represents the targeted instance of CIM_NumericSensor.

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

6.2.2.6.3.2 Pseudo Code

```
&lSetNumericSensorThreshold ( “UpperThresholdCritical”, <request value>,
      $instance.getObjectPath() );
&smEnd;
```

6.2.2.7 Set Command Form for the LowerThresholdFatal Property

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

6.2.2.7.1 Command Form

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

6.2.2.7.2 CIM Requirements

```
sint32 LowerThresholdFatal;
```

6.2.2.7.3 Behavior Requirements

6.2.2.7.3.1 Preconditions

`$instance` represents the targeted instance of CIM_NumericSensor.

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

6.2.2.7.3.2 Pseudo Code

```
&lSetNumericSensorThreshold ( “LowerThresholdFatal”, <request value>,
       $instance.getObjectPath() );
&smEnd;
```

6.2.2.8 Set Command Form for UpperThresholdFatal Property

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

6.2.2.8.1 Command Form

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

6.2.2.8.2 CIM Requirements

```
sint32 UpperThresholdFatal;
```

6.2.2.8.3 Behavior Requirements

6.2.2.8.3.1 Preconditions

`$instance` represents the targeted instance of CIM_NumericSensor.

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

6.2.2.8.3.2 Pseudo Code

```
&lSetNumericSensorThreshold ( “UpperThresholdFatal”, <request value>,
       $instance.getObjectPath() );
&smEnd;
```

6.2.2.9 Set Command Form for Restoring Threshold Defaults

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

6.2.2.9.1 Command Form

```
set –default <CIM_NumericSensor single object> SettableThresholds
```

6.2.2.9.2 CIM Requirements

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

6.2.2.9.3 Behavior Requirements

6.2.2.9.3.1 Preconditions

`$instance` represents the targeted instance of CIM_NumericSensor.

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

6.2.2.9.3.2 Pseudo Code

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

6.2.3 Show

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

6.2.3.1 Show Command Form for Multiple Objects Target

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

6.2.3.1.1 Command Form

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

6.2.3.1.2 CIM Requirements

See CIM_NumericSensor in the “CIM Elements” section of the *Sensors Profile* for the list of mandatory properties.

6.2.3.1.3 Behavior Requirements

6.2.3.1.3.1 Preconditions

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

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

6.2.3.1.3.2 Pseudo Code

```
#propertylist[] = NULL;
if ( false == #all)
   {
   #propertylist[] = <array of mandatory non-key property names (see CIM
      Requirements)>;
   }
&smShowInstances ( “CIM_NumericSensor”, “CIM_ElementCapabilities”,
   $containerInstance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.2.3.2 Show Command Form for a Single Object Target

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

6.2.3.2.1 Command Form

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

6.2.3.2.2 CIM Requirements

See CIM_NumericSensor in the “CIM Elements” section of the *Sensors Profile* for the list of mandatory properties.

6.2.3.2.3 Behavior Requirements

6.2.3.2.3.1 Preconditions

`$instance` represents the targeted instance of CIM_NumericSensor.

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

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

6.2.3.2.3.2 Pseudo Code

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

6.2.4 Start

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

6.2.4.1 Command Form

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

6.2.4.2 CIM Requirements

```
uint16 EnabledState;
uint16 RequestedState;
uint32 CIM_NumericSensor.RequestStateChange (
      [IN] uint16 RequestedState,
      [OUT] REF CIM_ConcreteJob Job,
      [IN] datetime TimeoutPeriod );
```

6.2.4.3 Behavior Requirements

6.2.4.3.1 Preconditions

`$instance` represents the targeted instance of CIM_NumericSensor.

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

6.2.4.3.2 Pseudo Code

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

6.2.5 Stop

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

6.2.5.1 Command Form

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

6.2.5.2 CIM Requirements

```
uint16 EnabledState;
uint16 RequestedState;
uint32 CIM_NumericSensor.RequestStateChange (
       [IN] uint16 RequestedState,
       [OUT] REF CIM_ConcreteJob Job,
       [IN] datetime TimeoutPeriod );
```

6.2.5.3 Behavior Requirements

6.2.5.3.1 Preconditions

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

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

6.2.5.3.2 Pseudo Code

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

6.3 CIM_EnabledLogicalElementCapabilities

The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in DSP0216.

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

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	

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

6.3.1 Show

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

6.3.1.1 Show Command Form for Multiple Objects Target

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

6.3.1.1.1 Command Form

`show <CIM_EnabledLogicalElementCapabilities multiple objects>`

6.3.1.1.2 CIM Requirements

See CIM_EnabledLogicalElementCapabilities in the “CIM Elements” section of the *Sensors Profile* for the list of mandatory properties.

6.3.1.1.3 Behavior Requirements

6.3.1.1.3.1 Preconditions

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

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

6.3.1.1.3.2 Pseudo Code

```
#propertylist[] = NULL;
if ( false == #all)
   {
   #propertylist[] = <array of mandatory non-key property names (see CIM
      Requirements)>;
   }
&smShowInstances ( “CIM_EnabledLogicalElementCapabilities”, “CIM_MemberOfCollection”,
   $containerInstance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.3.1.2 Show Command Form for Single Object Target

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

6.3.1.2.1 Command Form

```
show <CIM_EnabledLogicalElementCapabilities single object>
```

6.3.1.2.2 CIM Requirements

See CIM_EnabledLogicalElementCapabilities in the “CIM Elements” section of the *Sensors Profile* for the list of mandatory properties.

6.3.1.2.3 Behavior Requirements

6.3.1.2.3.1 Preconditions

`$instance` represents the targeted instance of CIM_EnabledLogicalElementCapabilities.

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

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

6.3.1.2.3.2 Pseudo Code

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

6.4 CIM_ElementCapabilities

The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in DSP0216.

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

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	

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

6.4.1 Ordering of Results

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

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

6.4.2 Show

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

6.4.2.1 Show Command Form for Multiple Objects Target – CIM_EnabledLogicalElementCapabilities Reference

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

6.4.2.1.1 Command Form

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

6.4.2.1.2 CIM Requirements

See CIM_ElementCapabilities in the “CIM Elements” section of the *Sensors Profile* for the list of mandatory properties.

6.4.2.1.3 Behavior Requirements

6.4.2.1.3.1 Preconditions

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

6.4.2.1.3.2 Pseudo Code

```
&smShowAssociationInstances ( “CIM_ElementCapabilities”, $instance.getObjectPath() );
&smEnd;
```

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

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

6.4.2.2.1 Command Form

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

6.4.2.2.2 CIM Requirements

See CIM_ElementCapabilities in the “CIM Elements” section of the *Sensors Profile* for the list of mandatory properties.

6.4.2.2.3 Behavior Requirements

6.4.2.2.3.1 Preconditions

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

6.4.2.2.3.2 Pseudo Code

```
&smShowAssociationInstances ( “CIM_ElementCapabilities”, $instance.getObjectPath(),
      #propertylist[] );
&smEnd;
```

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

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

6.4.2.3.1 Command Form

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

6.4.2.3.2 CIM Requirements

See CIM_ElementCapabilities in the “CIM Elements” section of the *Sensors Profile* for the list of mandatory properties.

6.4.2.3.3 Behavior Requirements

6.4.2.3.3.1 Preconditions

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

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

6.4.2.3.3.2 Pseudo Code

```
&smShowAssociationInstance ( “CIM_ElementCapabilities”, $instanceA.getObjectPath(),
      $instanceB.getObjectPath() );
&smEnd;
```

6.5 CIM_SystemDevice

The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in DSP0216.

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

6.5.2.3.3.2 Pseudo Code

```
&smShowAssociationInstance ( “CIM_SystemDevice”, $instanceA.getObjectPath(),
      $instanceB.getObjectPath() );
&smEnd;
```

6.6 CIM_AssociatedSensor

The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in DSP0216.

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

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	

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

6.6.1 Ordering of Results

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

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

6.6.2 Show

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

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

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

6.6.2.1.1 Command Form

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

6.6.2.1.2 CIM Requirements

See CIM_AssociatedSensor in the “CIM Elements” section of the *Sensors Profile* for the list of mandatory properties.

6.6.2.1.3 Behavior Requirements

6.6.2.1.3.1 Preconditions

`$instance` represents the instance of a subclass of CIM_ManagedSystemElement which is referenced by CIM_AssociatedSensor.

6.6.2.1.3.2 Pseudo Code

```
&smShowAssociationInstances ( “CIM_AssociatedSensor”, $instance.getObjectPath() );
&smEnd;
```

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

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

6.6.2.2.1 Command Form

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

6.6.2.2.2 CIM Requirements

See CIM_AssociatedSensor in the “CIM Elements” section of the *Sensors Profile* for the list of mandatory properties.

6.6.2.2.3 Behavior Requirements

6.6.2.2.3.1 Preconditions

`$instance` represents the instance of CIM_Sensor or CIM_NumericSensor which is referenced by CIM_AssociatedSensor.

6.6.2.2.3.2 Pseudo Code

```
&smShowAssociationInstances ( “CIM_AssociatedSensor”, $instance.getObjectPath() );
&smEnd;
```

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

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

6.6.2.3.1 Command Form

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

6.6.2.3.2 CIM Requirements

See CIM_AssociatedSensor in the “CIM Elements” section of the *Sensors Profile* for the list of mandatory properties.

6.6.2.3.3 Behavior Requirements

6.6.2.3.3.1 Preconditions

`$instanceA` represents the referenced instance of CIM_Sensor or CIM_NumericSensor through CIM_AssociatedSensor association.

`$instanceB` represents the instance of subclass of CIM_ManagedSystemElement which is referenced by CIM_AssociatedSensor.

6.6.2.3.3.2 Pseudo Code

```
&smShowAssociationInstance ( “CIM_AssociatedSensor”, $instanceA.getObjectPath(),
      $instanceB.getObjectPath() );
&smEnd;
```

ANNEX A
(informative)

Change Log

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