Document Number: DSP0809

Date: 2009-06-04

Version: 1.0.0

System Memory Profile SM CLP Mapping Specification

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

CONTENTS

Tables

56

Foreword

The *System Memory Profile SM CLP Mapping Specification* (DSP0809) 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
- Jeff Lynch – IBM
- Perry Vincent – Intel
- John Leung – Intel

Introduction

This document defines the SM CLP mapping for CIM elements described in the *System Memory 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 *System Memory Profile* using CIM operations.

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

System Memory Profile 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 *System Memory 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 DSP1026, *System Memory Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1026_1.0.pdf

DMTF DSP0216, *SM CLP-to-CIM Common Mapping Specification 1.0*,
http://www.dmtf.org/standards/published_documents/DSP0216_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.exe?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

The following symbols and abbreviations are used in this document.

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
Storage Management Initiative

4.7
SNIA
Storage Networking Industry Association

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 the *SM CLP-to-CIM Common Mapping Specification 1.0* (DSP0216).

- smShowInstance
- smShowInstances
- smShowAssociationInstance
- smShowAssociationInstances

This mapping does not define any recipes for local reuse.

6 Mappings

The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in the *System Memory Profile*.

6.1 CIM_ElementCapabilities

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_ElementCapabilities

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	

Command Verb	Requirement	Comments
Show	Shall	See 6.1.2
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`.

6.1.1 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.

6.1.2 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.

6.1.2.1 Show Command Form for a Single Instance Target – CIM_Memory 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_Memory.

6.1.2.1.1 Command Form

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

6.1.2.1.2 CIM Requirements

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

6.1.2.1.3 Behavior Requirements

6.1.2.1.3.1 Preconditions

In this section `$instance` represents the instance of CIM_Memory which is referenced by CIM_ElementCapabilities.

6.1.2.1.3.2 Pseudo Code

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

6.1.2.2 Show Command Form for Multiple Instances –CIM_EnabledLogicalElementCapabilities Reference

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

6.1.2.2.1 Command Form

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

6.1.2.2.2 CIM Requirements

See CIM_ElementCapabilities in the “CIM Elements” section of the *System Memory Profile* for the list of mandatory properties and CIM classes that can be referenced.

6.1.2.2.3 Behavior Requirements

6.1.2.2.3.1 Preconditions

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

6.1.2.2.3.2 Pseudo Code

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

6.1.2.3 Show Command Form for a Single Instance – 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.

6.1.2.3.1 Command Form

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

6.1.2.3.2 CIM Requirements

See CIM_ElementCapabilities in the “CIM Elements” section of the *System Memory Profile* for the list of mandatory properties and CIM classes that can be referenced.

6.1.2.3.3 Behavior Requirements

6.1.2.3.3.1 Preconditions

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

6.1.2.3.3.2 Pseudo Code

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

6.2 CIM_EnabledLogicalElementCapabilities

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_EnabledLogicalElementCapabilities

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.2.2.
Start	Not supported	
Stop	Not supported	

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

6.2.1 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_EnabledLogicalElementCapabilities are unordered; therefore, no algorithm is defined.

6.2.2 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.

6.2.2.1 Show Command Form for Multiple Instances Target

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

6.2.2.1.1 Command Form

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

6.2.2.1.2 CIM Requirements

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

6.2.2.1.3 Behavior Requirements

6.2.2.1.3.1 Preconditions

In this section `$containerInstance` represents the instance of CIM_ConcreteCollection, 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.

6.2.2.1.3.2 Pseudo Code

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

6.2.2.2 Show Command Form for a Single Instance Target

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

6.2.2.2.1 Command Form

`show <CIM_EnabledLogicalElementCapabilities single instance>`

6.2.2.2.2 CIM Requirements

See CIM_EnabledLogicalElementCapabilities in the "CIM Elements" section of the *System Memory Profile* for the list of mandatory properties.

6.2.2.2.3 Behavior Requirements

6.2.2.2.3.1 Preconditions

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

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

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

6.2.2.2.3.2 Pseudo Code

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

6.3 CIM_Memory

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.

314 **Table 3 – Command Verb Requirements for CIM_Memory**

Command Verb	Requirement	Comments
Create	Shall not	
Delete	Shall not	
Dump	Shall not	
Load	Shall not	
Reset	Shall not	
Set	May	See 6.3.2.
Show	Shall	See 6.3.3.
Start	Shall not	
Stop	Shall not	

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

317 ### 6.3.1 Ordering of Results

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

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

321 ### 6.3.2 Set

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

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

325 #### 6.3.2.1 General Usage of Set for a Single Property

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

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

331 ##### 6.3.2.1.1 Command Form

332 ```
set <CIM_Memory single instance> <propertyname>=<propertyvalue>
```

333 ##### 6.3.2.1.2 CIM Requirements

334 The ModifyInstance operation is required.

335 ##### 6.3.2.1.3 Behavior Requirements

336 ```
$instance=<CIM_Memory single instance>
337 #propertyNames[] = {<propertyname>};
338 #propertyValues[] = {<propertyvalue>};
339 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
340 &smEnd;
```

6.3.2.2 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 isn't 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 *System Memory Profile*.

6.3.2.2.1 Command Form

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

6.3.2.2.2 CIM Requirements

The ModifyInstance operation is required.

6.3.2.2.3 Behavior Requirements

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

6.3.3 Show

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

6.3.3.1 Show Command Form for Multiple Instances Target

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

6.3.3.1.1 Command Form

```
show <CIM_Memory multiple instances>
```

6.3.3.1.2 CIM Requirements

See CIM_Memory in the "CIM Elements" section of the *System Memory Profile* for the list of mandatory properties.

6.3.3.1.3 Behavior Requirements

6.3.3.1.3.1 Preconditions

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

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

6.3.3.1.3.2 Pseudo Code

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

6.3.3.2 Show Command Form for a Single Instance Target

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

6.3.3.2.1 Command Form

```
show <CIM_Memory single instance>
```

6.3.3.2.2 CIM Requirements

See CIM_Memory in the "CIM Elements" section of the *System Memory Profile* for the list of mandatory properties.

6.3.3.2.3 Behavior Requirements

6.3.3.2.3.1 Preconditions

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

```
$instance=<CIM_Memory single instance>;
```

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

6.3.3.2.3.2 Pseudo Code

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

6.4 CIM_SystemDevice

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.

415 **Table 4 – Command Verb Requirements for CIM_SystemDevice**

Command Verb	Requirement	Comments
Create	Shall not	
Delete	Shall not	
Dump	Shall not	
Load	Shall not	
Reset	Shall not	
Set	Shall not	
Show	Shall	See 6.4.2.
Start	Shall not	
Stop	Shall not	

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

418 ### 6.4.1 Ordering of Results

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

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

422 ### 6.4.2 Show

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

425 #### 6.4.2.1 Show Command Form for a Single Instance Target – CIM_ComputerSystem Reference

426 This command form is used to show many instances of CIM_SystemDevice. This command form
427 corresponds to a `show` command issued against the instance of CIM_SystemDevice where only one
428 reference is specified and the reference is to the scoping instance of CIM_ComputerSystem.

429 ##### 6.4.2.1.1 Command Form

430 ```
show <CIM_SystemDevice single instance>
```

431 ##### 6.4.2.1.2 CIM Requirements

432 See CIM_SystemDevice in the “CIM Elements” section of the *System Memory Profile* for the list of
433 mandatory properties and CIM classes that can be referenced.

434 ##### 6.4.2.1.3 Behavior Requirements

435 ###### 6.4.2.1.3.1 Preconditions

436 In this section `$instance` represents the instance of a CIM_ComputerSystem, which is referenced by
437 CIM_SystemDevice.
```

6.4.2.1.3.2 Pseudo Code

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

6.4.2.2 Show Command Form for a Single Instance – CIM_Memory Reference

This command form is used to show a single instance of CIM_SystemDevice. This command form corresponds to a `show` command issued against a single instance of CIM_SystemDevice, where only one reference is specified and the reference is to the instance of CIM_Memory.

6.4.2.2.1 Command Form

```
show <CIM_SystemDevice single instance>
```

6.4.2.2.2 CIM Requirements

See CIM_SystemDevice in the “CIM Elements” section of the *System Memory Profile* for the list of mandatory properties and CIM classes that can be referenced.

6.4.2.2.3 Behavior Requirements

6.4.2.2.3.1 Preconditions

In this section `$instance` represents the instance of CIM_Memory which is referenced by CIM_SystemDevice.

6.4.2.2.3.2 Pseudo Code

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

6.4.2.3 Show Command Form for a Single Instance – 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_SystemDevice where both references are specified and therefore the desired instance is unambiguously identified.

6.4.2.3.1 Command Form

```
show <CIM_SystemDevice single instance>
```

6.4.2.3.2 CIM Requirements

See CIM_SystemDevice in the “CIM Elements” section of the *System Memory Profile* for the list of mandatory properties and CIM classes that can be referenced.

6.4.2.3.3 Behavior Requirements

6.4.2.3.3.1 Preconditions

In this section `$instanceA` represents the referenced instance of CIM_Memory through CIM_SystemDevice association. `$instanceB` represents the instance of CIM_Memory which is referenced by CIM_SystemDevice.

6.4.2.3.3.2 Pseudo Code

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

476 # ANNEX A
477 (informative)
478
479
480 # Change Log

Version	Date	Author	Description
1.0.0	2009-06-04		DMTF Standard Release

481