Document Number: DSP0808

Date: 2009-06-04

Version: 1.0.0

CPU Profile SM CLP Mapping Specification

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

CONTENTS

Tables

Foreword

The *CPU Profile SM CLP Mapping Specification* (DSP0808) 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 DMTF 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 *CPU 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 *CPU Profile* using CIM operations.

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

CPU 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 *CPU 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 DSP1022, *CPU Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1022_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?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-S
Storage Management Initiative Specification

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).

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

6 Mappings

The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in the *CPU 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_AssociatedCacheMemory

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_AssociatedCacheMemory

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

Command Verb	Requirement	Comments
Reset	Not supported	
Set	Not supported	
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`, `load`, `reset`, `set`, `start`, and `stop`.

6.1.1 Ordering of Results

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

- Results for CIM_AssociatedCacheMemory 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_AssociatedCacheMemory. Implementations shall support the use of the `show` verb with CIM_AssociatedCacheMemory.

6.1.2.1 Show Command Form for Multiple Instances Target – CIM_Processor Reference

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

6.1.2.1.1 Command Form

```
show <CIM_AssociatedCacheMemory multiple instances>
```

6.1.2.1.2 CIM Requirements

See CIM_AssociatedCacheMemory in the “CIM Elements” section of the *CPU 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 a CIM_Processor, which is referenced by CIM_AssociatedCacheMemory.

6.1.2.1.3.2 Pseudo Code

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

6.1.2.2 Show Command Form for Multiple Instances Target – CIM_ProcessorCore Reference

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

6.1.2.2.1 Command Form

```
show <CIM_AssociatedCacheMemory multiple instances>
```

6.1.2.2.2 CIM Requirements

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

6.1.2.2.3 Behavior Requirements

6.1.2.2.3.1 Preconditions

In this section `$instance` represents the instance of a CIM_ProcessorCore, which is referenced by CIM_AssociatedCacheMemory.

6.1.2.2.3.2 Pseudo Code

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

6.1.2.3 Show Command Form for Multiple Instances Target – CIM_Memory Reference

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

6.1.2.3.1 Command Form

```
show <CIM_AssociatedCacheMemory multiple instances>
```

6.1.2.3.2 CIM Requirements

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

6.1.2.3.3 Behavior Requirements

6.1.2.3.3.1 Preconditions

In this section $instance represents the instance of a CIM_Memory, which is referenced by CIM_AssociatedCacheMemory.

6.1.2.3.3.2 Pseudo Code

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

6.1.2.4 Show Command Form for a Single Instance Target – Both References

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

6.1.2.4.1 Command Form

```
show <CIM_AssociatedCacheMemory single instance>
```

6.1.2.4.1.1 CIM Requirements

See CIM_AssociatedCacheMemory in the "CIM Elements" section of the *CPU Profile* for the list of mandatory properties.

6.1.2.4.2 Behavior Requirements

6.1.2.4.2.1 Preconditions

In this section `$instanceA` represents the referenced instance of CIM_Processor or CIM_ProcessorCore through CIM_AssociatedCacheMemory association. `$instanceB` represents the instance of CIM_Memory which is referenced by CIM_AssociatedCacheMemory.

6.1.2.4.2.2 Pseudo Code

```
&smShowAssociationInstance ( "CIM_AssociatedCacheMemory", $instanceA.getObjectPath(),
   $instanceB.getObjectPath() );
&smEnd;
```

6.2 CIM_ConcreteComponent

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_ConcreteComponent

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`, `load`, `reset`, `set`, `start`, and `stop`.

6.2.1 Ordering of Results

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

- Results for CIM_ConcreteComponent 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_ConcreteComponent. Implementations shall support the use of the `show` verb with CIM_ConcreteComponent.

6.2.2.1 Show Command Form for Multiple Instances Target – CIM_Processor Reference

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

6.2.2.1.1 Command Form

```
show <CIM_ConcreteComponent multiple instances>
```

6.2.2.1.2 CIM Requirements

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

6.2.2.1.3 Behavior Requirements

6.2.2.1.3.1 Preconditions

In this section `$instance` represents the instance of a CIM_Processor, which is referenced by CIM_ConcreteComponent.

6.2.2.1.3.2 Pseudo Code

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

6.2.2.2 Show Command Form for Multiple Instances Target – CIM_ProcessorCore Reference

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

6.2.2.2.1 Command Form

```
show <CIM_ConcreteComponent multiple instances>
```

6.2.2.2.2 CIM Requirements

See CIM_AssociatedCacheMemory in the “CIM Elements” section of the *CPU 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 instance of a CIM_ProcessorCore, which is referenced by CIM_ConcreteComponent.

6.2.2.2.3.2 Pseudo Code

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

6.2.2.3 Show Command Form for a Single Instance Target – CIM_HardwareThread Reference

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

6.2.2.3.1 Command Form

```
show <CIM_ConcreteComponent single instance>
```

6.2.2.3.2 CIM Requirements

See CIM_AssociatedCacheMemory in the "CIM Elements" section of the *CPU Profile* for the list of mandatory properties.

6.2.2.3.3 Behavior Requirements

6.2.2.3.3.1 Preconditions

In this section `$instance` represents the instance of CIM_HardwareThread which is referenced by CIM_ConcreteComponent.

6.2.2.3.3.2 Pseudo Code

```
&smShowAssociationInstances ( "CIM_ConcreteComponent", $instance.getObjectPath() );
&smEnd;
```

6.2.2.4 Show Command Form for a Single Instance Target – Both References: CIM_Processor and CIM_ProcessorCore

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

6.2.2.4.1 Command Form

```
show <CIM_ConcreteComponent single instance>
```

6.2.2.4.2 CIM Requirements

See CIM_AssociatedCacheMemory in the "CIM Elements" section of the *CPU Profile* for the list of mandatory properties.

6.2.2.4.3 Behavior Requirements

6.2.2.4.3.1 Preconditions

In this section `$instanceA` represents the referenced instance of CIM_Processor through CIM_ConcreteComponent association. `$instanceB` represents the instance of CIM_ProcessorCore which is referenced by CIM_ConcreteComponent.

6.2.2.4.3.2 Pseudo Code

```
&smShowAssociationInstance ( "CIM_ConcreteComponent", $instanceA.getObjectPath(),
   $instanceB.getObjectPath() );
&smEnd;
```

6.2.2.5 Show Command Form for a Single Instance Target – Both References: CIM_Processor and CIM_ProcessorCore

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

6.2.2.5.1 Command Form

```
show <CIM_ConcreteComponent single instance>
```

6.2.2.5.2 CIM Requirements

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

6.2.2.5.3 Behavior Requirements

6.2.2.5.3.1 Preconditions

In this section `$instanceA` represents the referenced instance of CIM_Processor through CIM_ConcreteComponent association. `$instanceB` represents the instance of CIM_ProcessorCore which is referenced by CIM_ConcreteComponent.

6.2.2.5.3.2 Pseudo Code

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

6.3 CIM_HardwareThread

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_HardwareThread

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	May	See 6.3.2.
Set	May	See 6.3.3.
Show	Shall	See 6.3.4.
Start	May	See 6.3.5.
Stop	May	See 6.3.6.

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

6.3.1 Ordering of Results

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

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

6.3.2 Reset

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

6.3.2.1 Command Form

```
reset <CIM_HardwareThread single instance>
```

6.3.2.1.1 CIM Requirements

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

6.3.2.1.2 Behavior Requirements

6.3.2.1.2.1 Preconditions

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

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

6.3.2.1.2.2 Pseudo Code

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

6.3.3 Set

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

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

6.3.3.1 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 CPU Profile.

6.3.3.1.1 Command Form

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

6.3.3.1.2 CIM Requirements

See CIM_HardwareThread in the “CIM Elements” section of the CPU Profile for the list of mandatory properties.

6.3.3.1.3 Behavior Requirements

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

6.3.3.2 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 CPU Profile.

6.3.3.2.1 Command Form

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

6.3.3.2.2 CIM Requirements

See CIM_HardwareThread in the “CIM Elements” section of the CPU Profile for the list of mandatory properties.

6.3.3.2.3 Behavior Requirements

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

6.3.4 Show

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

6.3.4.1 Show Command Form for Multiple Instances Target

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

6.3.4.1.1 Command Form

```
show <CIM_HardwareThread multiple instances>
```

6.3.4.1.2 CIM Requirements

See CIM_HardwareThread in the “CIM Elements” section of the CPU Profile for the list of mandatory properties.

6.3.4.1.3 Behavior Requirements

6.3.4.1.3.1 Preconditions

In this section `$containerInstance` represents the instance of CIM_ProcessorCore which represents the container system and is associated to the targeted instances of CIM_HardwareThread through the CIM_ConcreteComponent association.

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

6.3.4.1.3.2 Pseudo Code

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

6.3.4.2 Show Command Form for a Single Instance Target

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

6.3.4.2.1 Command Form

show <CIM_HardwareThread *single instance*>

6.3.4.2.2 CIM Requirements

See CIM_HardwareThread in the “CIM Elements” section of the CPU Profile for the list of mandatory properties.

6.3.4.2.3 Behavior Requirements

6.3.4.2.3.1 Preconditions

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

`$instance=<CIM_HardwareThread *single instance*>;`

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

6.3.4.2.3.2 Pseudo Code

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

6.3.5 Start

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

6.3.5.1.1 Command Form

```
start <CIM_HardwareThread single instance>
```

6.3.5.1.2 CIM Requirements

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

6.3.5.1.2.1 Behavior Requirements

6.3.5.1.2.2 Preconditions

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

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

6.3.5.1.2.3 Pseudo Code

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

6.3.6 Stop

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

6.3.6.1.1 Command Form

```
stop <CIM_HardwareThread single instance>
```

6.3.6.1.2 CIM Requirements

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

6.3.6.1.3 Behavior Requirements

6.3.6.1.3.1 Preconditions

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

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

6.3.6.1.3.2 Pseudo Code

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

6.4 CIM_Memory

The CLP implementation shall expose only a unique path for each instance of CIM_Memory. The association class for the path shall be CIM_AssociatedCacheMemory.

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_Memory

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	May	See 6.4.2.
Set	May	See 6.4.3.
Show	Shall	See 6.4.4.
Start	May	See 6.4.5.
Stop	May	See 6.4.6.

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

6.4.1 Ordering of Results

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

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

6.4.2 Reset

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

6.4.2.1.1 Command Form

```
reset <CIM_Memory single instance>
```

6.4.2.1.2 CIM Requirements

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

6.4.2.1.3 Behavior Requirements

6.4.2.1.3.1 Preconditions

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

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

6.4.2.1.3.2 Pseudo Code

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

6.4.3 Set

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

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

6.4.3.1 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 *CPU Profile*.

6.4.3.1.1 Command Form

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

6.4.3.1.2 CIM Requirements

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

6.4.3.1.3 Behavior Requirements

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

6.4.3.2 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 CPU Profile.

6.4.3.2.1 Command Form

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

6.4.3.2.2 CIM Requirements

See CIM_Memory in the “CIM Elements” section of the CPU Profile for the list of mandatory properties.

6.4.3.2.3 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;
```

6.4.4 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.

6.4.4.1 Show Command Form for Multiple Instances Target

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

6.4.4.1.1 Command Form

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

6.4.4.1.2 CIM Requirements

See CIM_Memory in the “CIM Elements” section of the CPU Profile for the list of mandatory properties.

6.4.4.1.3 Behavior Requirements

6.4.4.1.3.1 Preconditions

In this section $containerInstance represents the instance of CIM_Processor or CIM_ProcessorCore which represents the processor or the core that utilizes the cache memory and is associated to the targeted instances of CIM_Memory through the CIM_AssociatedCacheMemory association.

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

6.4.4.1.3.2 Pseudo Code

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

```
&smShowInstances ( “CIM_Memory”, “CIM_AssociatedCacheMemory” , NULL, NULL,
   $containerInstance.getObjectPath(), NULL, NULL, #propertylist[] );
&smEnd;
```

6.4.4.2 Show Command Form for a Single Instance Target

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

6.4.4.2.1 Command Form

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

6.4.4.2.2 CIM Requirements

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

6.4.4.2.3 Behavior Requirements

6.4.4.2.3.1 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.

6.4.4.2.3.2 Pseudo Code

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

6.4.5 Start

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

6.4.5.1.1 Command Form

```
start <CIM_Memory single instance>
```

6.4.5.1.2 CIM Requirements

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

6.4.5.1.3 Behavior Requirements

6.4.5.1.3.1 Preconditions

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

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

6.4.5.1.3.2 Pseudo Code

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

6.4.6 Stop

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

6.4.7 Command Form

```
stop <CIM_Memory single instance>
```

6.4.7.1.1 CIM Requirements

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

6.4.7.1.2 Behavior Requirements

6.4.7.1.2.1 Preconditions

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

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

6.4.7.1.2.2 Pseudo Code

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

6.5 CIM_ElementCapabilities

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.

Table 5 – 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 6.5.2.

Command Verb	Requirement	Comments
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`.

6.5.1 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.

6.5.2 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.

6.5.2.1 Show Command Form for Multiple Instances 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 an instance of CIM_EnabledLogicalElementCapabilities.

6.5.2.1.1 Command Form

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

6.5.2.1.2 CIM Requirements

See CIM_ElementCapabilities in the "CIM Elements" section of the *CPU Profile* for the list of mandatory properties.

6.5.2.1.3 Behavior Requirements

6.5.2.1.3.1 Preconditions

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

6.5.2.1.3.2 Pseudo Code

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

6.5.2.2 Show Command Form for Multiple Instances Target – CIM_ProcessorCapabilities 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 an instance of CIM_ProcessorCapabilities.

6.5.2.2.1 Command Form

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

6.5.2.2.2 CIM Requirements

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

6.5.2.2.3 Behavior Requirements

6.5.2.2.3.1 Preconditions

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

6.5.2.2.3.2 Pseudo Code

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

6.5.2.3 Show Command Form for a Single Instance – CIM_Processor 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_Processor.

6.5.2.3.1 Command Form

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

6.5.2.3.2 CIM Requirements

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

6.5.2.3.3 Behavior Requirements

6.5.2.3.3.1 Preconditions

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

6.5.2.3.3.2 Pseudo Code

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

6.5.2.4 Show Command Form for a Single Instance – CIM_ProcessorCore 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_ProcessorCore.

6.5.2.4.1 Command Form

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

6.5.2.4.2 CIM Requirements

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

6.5.2.4.3 Behavior Requirements

6.5.2.4.3.1 Preconditions

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

6.5.2.4.3.2 Pseudo Code

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

6.5.2.5 Show Command Form for a Single Instance – CIM_HardwareThread 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_HardwareThread.

6.5.2.5.1 Command Form

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

6.5.2.5.2 CIM Requirements

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

6.5.2.5.3 Behavior Requirements

6.5.2.5.3.1 Preconditions

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

6.5.2.5.3.2 Pseudo Code

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

6.5.2.6 Show Command Form for a Single Instance – 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.

6.5.2.6.1 Command Form

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

6.5.2.6.2 CIM Requirements

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

6.5.2.6.3 Behavior Requirements

6.5.2.6.3.1 Preconditions

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

6.5.2.6.3.2 Pseudo Code

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

6.5.2.7 Show Command Form for a Single Instance Target – Both References: CIM_ProcessorCapabilities and CIM_Processor

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

6.5.2.7.1 Command Form

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

6.5.2.7.2 CIM Requirements

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

6.5.2.7.3 Behavior Requirements

6.5.2.7.3.1 Preconditions

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

6.5.2.7.3.2 Pseudo Code

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

6.5.2.8 Show Command Form for a Single Instance Target – Both References: CIM_EnabledLogicalElementCapabilities and CIM_ProcessorCore

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

6.5.2.8.1 Command Form

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

6.5.2.8.2 CIM Requirements

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

6.5.2.8.3 Behavior Requirements

6.5.2.8.3.1 Preconditions

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

6.5.2.8.3.2 Pseudo Code

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

6.5.2.9 Show Command Form for a Single Instance Target – Both References: CIM_EnabledLogicalElementCapabilities and CIM_HardwareThread

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

6.5.2.9.1 Command Form

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

6.5.2.9.2 CIM Requirements

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

6.5.2.9.3 Behavior Requirements

6.5.2.9.3.1 Preconditions

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

6.5.2.9.3.2 Pseudo Code

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

6.5.2.10 Show Command Form for a Single Instance Target – Both References: CIM_EnabledLogicalElementCapabilities and CIM_Memory

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

6.5.2.10.1 Command Form

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

6.5.2.10.2 CIM Requirements

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

6.5.2.10.3 Behavior Requirements

6.5.2.10.3.1 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.

6.5.2.10.3.2 Pseudo Code

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

6.6 CIM_EnabledLogicalElementCapabilities

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_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 6.6.2.
Start	Not supported	
Stop	Not supported	

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

6.6.1 Ordering of Results

When results are returned for multiple instances of CIM_EnabledLogicalElementCapabilities, 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.

6.6.2 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.

6.6.2.1 Show Command Form for Multiple Instances Target

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

6.6.2.1.1 Command Form

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

6.6.2.1.2 CIM Requirements

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

6.6.2.1.3 Behavior Requirements

6.6.2.1.3.1 Preconditions

In this section `$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.

6.6.2.1.3.2 Pseudo Code

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

6.6.2.2 Show Command Form for a Single Instance Target

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

6.6.2.2.1 Command Form

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

6.6.2.2.2 CIM Requirements

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

6.6.2.2.3 Behavior Requirements

6.6.2.2.3.1 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.

6.6.2.2.3.2 Pseudo Code

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

6.7 CIM_Processor

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

Table 7 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 7 is for informational purposes only; in case of a conflict between Table 7 and requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 7.

Table 7 – Command Verb Requirements for CIM_Processor

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	May	See 6.7.2.
Set	May	See 6.7.3.
Show	Shall	See 6.7.4.
Start	May	See 6.7.5.
Stop	May	See 6.7.6.

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

6.7.1 Ordering of Results

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

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

6.7.2 Reset

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

6.7.2.1 Command Form

```
reset <CIM_Processor single instance>
```

6.7.2.2 CIM Requirements

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

6.7.2.3 Behavior Requirements

6.7.2.3.1 Preconditions

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

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

6.7.2.3.2 Pseudo Code

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

6.7.3 Set

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

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

6.7.3.1 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 *CPU Profile*.

6.7.3.2 Command Form

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

6.7.3.3 CIM Requirements

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

6.7.3.4 Behavior Requirements

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

6.7.3.5 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 *CPU Profile*.

6.7.3.5.1 Command Form

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

6.7.3.5.2 CIM Requirements

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

6.7.3.5.3 Behavior Requirements

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

6.7.4 Show

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

6.7.4.1 Show Command Form for Multiple Instances Target

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

6.7.4.1.1 Command Form

```
show <CIM_Processor multiple instances>
```

6.7.4.1.2 CIM Requirements

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

6.7.4.1.3 Behavior Requirements

6.7.4.1.3.1 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_Processor through the CIM_SystemDevice association.

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

6.7.4.1.3.2 Pseudo Code

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

6.7.4.2 Show Command Form for a Single Instance Target

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

6.7.4.2.1 Command Form

```
show <CIM_Processor single instance>
```

6.7.4.2.2 CIM Requirements

See CIM_Processor in the "CIM Elements" section of the *CPU Profile* for the list of mandatory properties.

6.7.4.2.3 Behavior Requirements

6.7.4.2.3.1 Preconditions

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

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

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

6.7.4.2.3.2 Pseudo Code

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

6.7.5 Start

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

6.7.5.1 Command Form

```
start <CIM_Processor single instance>
```

6.7.5.2 CIM Requirements

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

6.7.5.3 Behavior Requirements

6.7.5.3.1 Preconditions

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

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

6.7.5.3.2 Pseudo Code

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

6.7.6 Stop

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

6.7.6.1 Command Form

```
stop <CIM_Processor single instance>
```

6.7.6.2 CIM Requirements

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

6.7.6.3 Behavior Requirements

6.7.6.3.1 Preconditions

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

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

6.7.6.3.2 Pseudo Code

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

6.8 CIM_ProcessorCapabilities

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

Table 8 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 8 is for informational purposes only; in case of a conflict between Table 8 and requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 8.

Table 8 – Command Verb Requirements for CIM_ProcessorCapabilities

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 6.8.2.
Start	Not supported	
Stop	Not supported	

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

6.8.1 Ordering of Results

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

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

6.8.2 Show

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

6.8.2.1 Show Command Form for Multiple Instances Target

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

6.8.2.1.1 Command Form

```
show <CIM_ProcessorCapabilities multiple instances>
```

6.8.2.1.2 CIM Requirements

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

6.8.2.1.3 Behavior Requirements

6.8.2.1.3.1 Preconditions

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

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

6.8.2.1.3.2 Pseudo Code

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

6.8.2.2 Show Command Form for a Single Instance Target

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

6.8.2.2.1 Command Form

```
show <CIM_ProcessorCapabilities single instance>
```

6.8.2.2.2 CIM Requirements

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

6.8.2.2.3 Behavior Requirements

6.8.2.2.3.1 Preconditions

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

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

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

6.8.2.2.3.2 Pseudo Code

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

6.9 CIM_ProcessorCore

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

Table 9 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 9 is for informational purposes only; in case of a conflict between Table 9 and requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 9.

Table 9 – Command Verb Requirements for CIM_ProcessorCore

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	May	See 6.9.2.
Set	May	See 6.9.3.
Show	Shall	See 6.9.4.
Start	May	See 6.9.5.
Stop	May	See 6.9.6.

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

6.9.1 Ordering of Results

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

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

6.9.2 Reset

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

6.9.2.1 Command Form

```
reset <CIM_ProcessorCore single instance>
```

6.9.2.2 CIM Requirements

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

6.9.2.3 Behavior Requirements

6.9.2.3.1.1 Preconditions

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

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

6.9.2.3.1.2 Pseudo Code

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

6.9.3 Set

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

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

6.9.3.1 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 *CPU Profile*.

6.9.3.1.1 Command Form

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

6.9.3.1.2 CIM Requirements

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

6.9.3.1.3 Behavior Requirements

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

6.9.3.2 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 *CPU Profile*.

6.9.3.2.1 Command Form

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

6.9.3.2.2 CIM Requirements

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

6.9.3.2.3 Behavior Requirements

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

6.9.4 Show

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

6.9.4.1 Show Command Form for Multiple Instances Target

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

6.9.4.1.1 Command Form

```
show <CIM_ProcessorCore multiple instances>
```

6.9.4.1.2 CIM Requirements

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

6.9.4.1.3 Behavior Requirements

6.9.4.1.3.1 Preconditions

In this section `$containerInstance` represents the instance of CIM_Processor which represents the container system and is associated to the targeted instances of CIM_ProcessorCore through the CIM_ConcreteComponent association.

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

6.9.4.1.3.2 Pseudo Code

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

6.9.4.2 Show Command Form for a Single Instance Target

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

6.9.4.2.1 Command Form

```
show <CIM_ProcessorCore single instance>
```

6.9.4.2.2 CIM Requirements

See CIM_ProcessorCore in the "CIM Elements" section of the *CPU Profile* for the list of mandatory properties.

6.9.4.2.3 Behavior Requirements

6.9.4.2.3.1 Preconditions

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

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

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

6.9.4.2.3.2 Pseudo Code

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

6.9.5 Start

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

6.9.5.1 Command Form

```
start <CIM_ProcessorCore single instance>
```

6.9.5.2 CIM Requirements

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

6.9.5.3 Behavior Requirements

6.9.5.3.1.1 Preconditions

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

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

6.9.5.3.1.2 Pseudo Code

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

6.9.6 Stop

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

6.9.6.1 Command Form

```
stop <CIM_ProcessorCore single instance>
```

6.9.6.1.1 CIM Requirements

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

6.9.6.1.2 Behavior Requirements

6.9.6.1.2.1 Preconditions

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

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

6.9.6.1.2.2 Pseudo Code

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

6.10 CIM_SystemDevice

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

Table 10 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 10 is for informational purposes only; in case of a conflict between Table 10 and requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 10.

Table 10 – 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 6.10.2.
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`.

6.10.1 Ordering of Results

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

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

6.10.2 Show

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

6.10.2.1 Show Command Form for Multiple Instances Target – CIM_ComputerSystem Reference

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

6.10.2.1.1 Command Form

```
show <CIM_SystemDevice multiple instances>
```

6.10.2.1.2 CIM Requirements

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

6.10.2.1.3 Behavior Requirements

6.10.2.1.3.1 Preconditions

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

6.10.2.1.3.2 Pseudo Code

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

6.10.2.2 Show Command Form for a Single Instance Target – CIM_Processor 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_Processor.

6.10.2.2.1 Command Form

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

6.10.2.2.2 CIM Requirements

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

6.10.2.2.3 Behavior Requirements

6.10.2.2.3.1 Preconditions

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

6.10.2.2.3.2 Pseudo Code

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

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

6.10.2.3.1 Command Form

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

6.10.2.3.2 CIM Requirements

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

6.10.2.3.3 Behavior Requirements

6.10.2.3.3.1 Preconditions

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

6.10.2.3.3.2 Pseudo Code

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

ANNEX A
(informative)

Change Log

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