Document Number: DSP0807

Date: 2009-06-04

Version: 1.0.0

Pass-Through Module Profile SM CLP Command Mapping Specification

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

CONTENTS

Tables

Foreword

The *Pass-Through Module Profile SM CLP Command Mapping Specification* (DSP0807) 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:

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

Introduction

This document defines the SM CLP mapping for CIM elements described in the *Pass-Through Module 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 *Pass-Through Module Profile* using CIM operations.

The target audience for this specification is implementers of the SM CLP support for the *Pass-Through Module Profile*.

Pass-Through Module Profile SM CLP Command 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 *Pass-Through Module 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 DSP1020, *Pass-Through Module Profile 1.0*, http://www.dmtf.org/standards/published_documents/DSP1020_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

4.8
UFsT
User Friendly selection Tag

5 Recipes

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

- smStartRSC
- smStopRSC
- smResetRSC
- smShowInstance
- smShowInstances
- smSetInstance
- smShowAssociationInstances
- smShowAssociationInstance
- smMakeCommandStatus
- smAddError
- smCommandCompleted

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 *Pass-Through Module Profile*.

6.1 CIM_ElementCapabilities

The `cd` and `help` 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 instances of 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	
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_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.

The `show` command is used to display information about the CIM_ElementCapabilities instance or instances.

6.1.2.1 Show Multiple Instances – CIM_EnabledLogicalElementCapabilities Reference

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

6.1.2.1.1 Command Form

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

6.1.2.1.2 CIM Requirements

See the “CIM Elements” section of the *Pass-Through Module Profile*.

6.1.2.1.3 Behavior Requirements

6.1.2.1.3.1 Preconditions

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

6.1.2.1.3.2 Pseudo Code

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

6.1.2.2 Show a Single Instance – CIM_PassThroughModule Reference

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 the reference specified is to an instance of CIM_PassThroughModule. A single instance of CIM_EnabledLogicalElementCapabilities can be associated with each instance of a CIM_PassThroughModule. Therefore, a single instance will be returned.

6.1.2.2.1 Command Form

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

6.1.2.2.2 CIM Requirements

See the “CIM Elements” section of the *Pass-Through Module Profile*.

6.1.2.2.3 Behavior Requirements

6.1.2.2.3.1 Preconditions

`$instance` contains the instance of CIM_PassThroughModule which is referenced by CIM_ElementCapabilities.

6.1.2.2.3.2 Pseudo Code

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

6.1.2.3 Show 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 the “CIM Elements” section of the *Pass-Through Module Profile*.

6.1.2.3.3 Behavior Requirements

6.1.2.3.3.1 Preconditions

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

`$instanceB` contains the instance of CIM_PassThroughModule which is referenced by CIM_ElementCapabilities.

6.1.2.3.3.2 Pseudo Code

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

6.2 CIM_EnabledLogicalElementCapabilities

The `cd` and `help` 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 instances of 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`, `load`, `reset`, `set`, `start`, and `stop`.

6.2.1 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.

6.2.2 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.

The `show` verb is used to display information about an instance or instances of the CIM_EnabledLogicalElementCapabilities class.

6.2.2.1 Show a Single Instance

This command form is for the `show` verb applied to a single instance of CIM_EnabledLogicalElementCapabilities.

6.2.2.1.1 Command Form

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

6.2.2.1.2 CIM Requirements

See the "CIM Elements" section of the *Pass-Through Module Profile*.

6.2.2.1.3 Behavior Requirements

6.2.2.1.3.1 Preconditions

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

6.2.2.1.3.2 Pseudo Code

```
$instance=<CIM_EnabledLogicalElementCapabilities single instance>
#propertylist[] = NULL;
if ( false == #all) {
   #propertylist[] = {//all mandatory non-key properties}
}
&smShowInstance ( $instance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.2.2.2 Show Multiple Instances

This command form is for the `show` verb applied to multiple instances of CIM_EnabledLogicalElementCapabilities. This command form corresponds to UFsT-based selection within a capabilities collection.

6.2.2.2.1 Command Form

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

6.2.2.2.2 CIM Requirements

See the "CIM Elements" section of the *Pass-Through Module Profile*.

6.2.2.2.3 Behavior Requirements

6.2.2.2.3.1 Preconditions

`$containerInstance` contains the instance of CIM_ConcreteCollection for which we are displaying contained CIM_Capabilities instances CIM_Capabilities instances are addressed via the aggregating instance of CIM_ConcreteCollection.

`#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[] = {//all mandatory non-key properties}
}
&smShowInstances ( "CIM_EnabledLogicalElementCapabilities", "CIM_MemberOfCollection",
   $containerInstance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.3 CIM_PassThroughModule

The `cd` and `help` 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_PassThroughModule

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_PassThroughModule, implementations shall utilize the following algorithm to produce the natural (that is, default) ordering:

- Results for CIM_PassThroughModule 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_PassThroughModule. Implementations may support the use of the `reset` verb with CIM_PassThroughModule.

The `reset` verb is used to initiate a reset of the CIM_PassThroughModule.

6.3.2.1 Reset a Single Instance

This command form is for the initiation of a reset action against a single pass-through module. The mapping is implemented as an invocation of the RequestStateChange( ) method on the instance.

6.3.2.1.1 Command Form

```
reset <CIM_PassThroughModule single instance>
```

6.3.2.1.2 CIM Requirements

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

6.3.2.1.3 Behavior Requirements

```
$instance=<CIM_PassThroughModule single instance>
smResetRSC ( $instance.getObjectPath() );
&smEnd;
```

6.3.3 Set

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

No properties of the CIM_PassThroughModule instance are writeable via the intrinsic ModifyInstance operation. Therefore, the only command form specified is for requesting a state change on the instance via assignment to the RequestedState property.

6.3.3.1 Using Set to Modify Port Assignments

In this command form the `set` verb is used to modify the mapping between the specified internal and external ports. Each property name corresponds to a parameter of the AssignPorts( ) method and the property value corresponds to the desired value for the parameter.

6.3.3.1.1 Command Form

```
set <CIM_PassThroughModule Single Instance> mapped=<mapped> internalport=<internal
      port number> externalport=<external port number>
```

6.3.3.1.2 CIM Requirements

```
uint32 AssignPorts(
   [IN] boolean Mapped,
   [IN] uint16 InternalPort,
   [IN] uint16 ExternalPort);
```

6.3.3.1.3 Behavior Requirements

```
//Input parameters to method come straight from command line
#mapped=mapped
#InternalPort=internalport
#ExternalPort=externalport

$instance=<CIM_PassThroughModule single instance>
%InArguments[] = {newArgument("Mapped", #Mapped),
                  newArgument("InternalPort", #InternalPort),
                  newArgument("ExternalPort", #ExternalPort)}
%OutArguments[] = { }
#Error = InvokeMethod ($instance.getObjectPath(),
                       "AssignPorts",
                       %InArguments[],
                       %OutArguments[],
                       #returnStatus);
   if (0 != #Error.code)  {
      //method invocation failed
      if ( (null != #Error.$error) && (null != #Error.$error[0]) )    {
         //if the method invocation contains an embedded error
         //use it for the Error for the overall job
            &smAddError($job, #Error.$error[0]);
            &smMakeCommandStatus($job);
            &smEnd;
      }
      else if ( 17 == #Error.code ) {
         //17 – CIM_ERR_METHOD_NOT_FOUND
         // The specified extrinsic method does not exist.
         $OperationError = smNewInstance("CIM_Error");
         // CIM_ERR_METHOD_NOT_FOUND
         $OperationError.CIMStatusCode = 17;
         //Software Error
         $OperationError.ErrorType = 10;
         //Unknown
         $OperationError.PerceivedSeverity = 0;
         $OperationError.OwningEntity = DMTF:SMCLP;
         $OperationError.MessageID = 0x00000001;
         $OperationError.Message = "Operation is not supported."
         &smAddError($job, $OperationError);
         &smMakeCommandStatus($job);
         &smEnd;
      }
      else {
         //operation failed, but no detailed error instance, need to make one
         //make an Error instance and associate with job for Operation
            $OperationError = smNewInstance("CIM_Error");
            //CIM_ERR_FAILED
            $OperationError.CIMStatusCode = 1;
            //Software Error
            $OperationError.ErrorType = 4;
            //Unknown
            $OperationError.PerceivedSeverity = 0;
            $OperationError.OwningEntity = DMTF:SMCLP;
            $OperationError.MessageID = 0x00000009;
            $OperationError.Message = “An internal software error has occurred.”;
            &smAddError($job, $OperationError);
            &smMakeCommandStatus($job);
            &smEnd;
      }
   }//if CIM op failed
   else if (0 == #returnStatus)  {
      //completed successfully
      &smCommandCompleted($job);
      &smEnd;
   }
   else if (2 == #returnStatus)  {
      //generic failure
      $OperationError = smNewInstance(“CIM_Error”);
      //CIM_ERR_FAILED
      $OperationError.CIMStatusCode = 1;
      //Other
      $OperationError.ErrorType = 1;
      //Low
      $OperationError.PerceivedSeverity = 2;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000002;
      $OperationError.Message = “Failed. No further information is available.”;
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
   }
   else   {
      //unspecified return code, generic failure
      $OperationError = smNewInstance(“CIM_Error”);
      //CIM_ERR_FAILED
      $OperationError.CIMStatusCode = 1;
      //Other
      $OperationError.ErrorType = 1;
      //Low
      $OperationError.PerceivedSeverity = 2;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000002;
      $OperationError.Message = “Failed. No further information is available.”;
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   }
```

6.3.3.2 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 *Pass-Through Module Profile*.

6.3.3.2.1 Command Form

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

6.3.3.2.2 CIM Requirements

Any modifiable property. See the “CIM Elements” section of the *Pass-Through Module Profile*.

6.3.3.2.3 Behavior Requirements

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

6.3.3.3 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 *Pass-Through Module Profile*.

6.3.3.3.1 Command Form

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

6.3.3.3.2 CIM Requirements

See supported properties list above.

6.3.3.3.3 Behavior Requirements

```
$instance=<CIM_PassThroughModule 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_PassThroughModule. Implementations shall support the use of the `show` verb with CIM_PassThroughModule.

The `show` verb is used to display information about the pass-through module.

6.3.4.1 Show a Single Instance

This command form is for the `show` verb applied to a single instance of CIM_PassThroughModule.

6.3.4.1.1 Command Form

```
show <CIM_PassThroughModule single instance>
```

6.3.4.1.2 CIM Requirements

6.3.4.1.3 Behavior Requirements

6.3.4.1.3.1 Preconditions

`#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[] = {//all mandatory non-key properties };
}
$instance=<CIM_PassThroughModule single instance>
&smShowInstance ( $instance.getObjectPath(),#propertylist[] );
&smEnd;
```

6.3.4.2 Show Multiple Instances

This command form is for the `show` verb applied to multiple instances of CIM_PassThroughModule. This command form corresponds to UFsT-based selection within a scoping system.

6.3.4.2.1 Command Form

```
show <CIM_PassThroughModule multiple instances>
```

6.3.4.2.2 Behavior Requirements

6.3.4.2.2.1 Preconditions

`$containerInstance` contains the instance of CIM_ComputerSystem for which we are displaying scoped pass-through modules (CIM_PassThroughModule instances). The Pass-through module Profile requires the CIM_PassThroughModule instance be associated with its scoping system via an instance of the CIM_SystemDevice association.

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

6.3.4.2.2.2 Pseudo Code

```
#propertylist[] = NULL;
if ( false == #all) {
   #propertylist[] = {//all mandatory non-key properties };
}
&smShowInstances ( “CIM_PassThroughModule”, “CIM_SystemDevice”,
   $containerInstance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.3.5 Start

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

The `start` verb is used to enable a pass-through module.

6.3.5.1 Start a Single Instance

This command form is for the `start` verb applied to a single instance of CIM_PassThroughModule.

6.3.5.1.1 Command Form

```
start <CIM_PassThroughModule single instance>
```

6.3.5.1.2 CIM Requirements

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

6.3.5.1.3 Behavior Requirements

```
$instance=<CIM_PassThroughModule single instance>
smStartRSC ( $instance.getObjectPath() );
&smEnd;
```

6.3.6 Stop

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

The `stop` verb is used to disable a pass-through module.

6.3.6.1 Stop a Single Instance

This command form is for the `stop` verb applied to a single instance of CIM_PassThroughModule.

6.3.6.1.1 Command Form

```
stop <CIM_PassThroughModule single instance>
```

6.3.6.1.2 CIM Requirements

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

6.3.6.1.3 Behavior Requirements

```
$instance=<CIM_PassThroughModule single instance>
smStopRSC ( $instance.getObjectPath() );
&smEnd;
```

6.4 CIM_SystemDevice

The `cd` and `help` 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_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 6.4.1.
start	Shall not	
stop	Shall not	

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

6.4.1 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.

The `show` command is used to display information about the CIM_SystemDevice instance or instances.

6.4.1.1 Show Multiple Instances – CIM_ComputerSystem Reference

This command form is for the `show` verb applied to multiple instances. This command form corresponds to a `show` command issued against CIM_SystemDevice where only one reference is specified and the reference is to an instance of CIM_ComputerSystem.

6.4.1.1.1 Command Form

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

6.4.1.1.2 CIM Requirements

6.4.1.1.3 Behavior Requirements

6.4.1.1.3.1 Preconditions

`$instance` contains the instance of CIM_ComputerSystem which is referenced by CIM_SystemDevice.

6.4.1.1.3.2 Pseudo Code

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

6.4.1.2 Show a Single Instance – CIM_PassThroughModule Reference

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 the reference specified is to an instance of CIM_PassThroughModule. An instance of CIM_PassThroughModule is referenced by exactly one instance of CIM_SystemDevice. Therefore, a single instance will be returned.

6.4.1.2.1 Command Form

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

6.4.1.2.2 CIM Requirements

6.4.1.2.3 Behavior Requirements

6.4.1.2.3.1 Preconditions

`$instance` contains the instance of CIM_PassThroughModule which is referenced by CIM_SystemDevice.

6.4.1.2.3.2 Pseudo Code

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

6.4.1.3 Show 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.

6.4.1.3.1 Command Form

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

6.4.1.3.2 CIM Requirements

6.4.1.3.3 Behavior Requirements

6.4.1.3.3.1 Preconditions

`$instanceA` contains the instance of CIM_ComputerSystem which is referenced by CIM_SystemDevice.

`$instanceB` contains the instance of CIM_PassThroughModule which is referenced by CIM_SystemDevice.

6.4.1.3.3.2 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