Document Number: DSP0823

Date: 2009-07-14

Version: 1.0.0

Power State Management Profile to SM CLP Mapping Specification

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

CONTENTS

Tables

Foreword

The *Power State Management Profile to SM CLP Mapping Specification* (DSP0823) 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:

- RadhaKrishna R. Dasari – Dell
- Khachatur Papanyan – Dell
- Aaron Merkin – IBM
- Perry Vincent – Intel

Introduction

This document defines the SM CLP mapping for CIM elements described in the *Power State Management 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 *Power State Management Profile* using CIM operations.

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

Power State Management Profile to SM CLP Mapping Specification

1 Scope

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

2 Normative References

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

2.1 Approved References

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

DMTF DSP1027, *Power State Management Profile 1.0*, http://www.dmtf.org/standards/published_documents/DSP1027_1.0.pdf

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

2.2 Other References

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

3 Terms and Definitions

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

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

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

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

3.4

mandatory

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

3.5

may

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

3.6

need not

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

3.7

optional

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

3.8

shall

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

3.9

shall not

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

3.10

should

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

3.11

should not

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

4 Symbols and Abbreviated Terms

The following symbols and abbreviations are used in this document.

4.1

CIM

Common Information Model

4.2

CLP

Command Line Protocol

4.3

DMTF

Distributed Management Task Force

4.4
SM
Server Management

4.5
SMI-S
Storage Management Initiative Specification

4.6
SNIA
Storage Networking Industry Association

4.7
UFsT
User Friendly Selection Tag

5 Recipes

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

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

The following section details the Local Recipe defined for use in this mapping.

5.1 smRequestPowerStateChange

5.1.1 Description

This method is used to change the power state of a computer system.

5.1.2 Pseudo Code

```
sub void smRequestPowerStateChange (uint64 #PowerState, $targetSystem->,#time,
      $targetPMS->)
{
//PowerState parameter contains the requested power state.
//$targetsystem-> parameter contains the target system whose power state
// needs to be set.
// #time parameter contains the time at which the request needs to be
// performed.
// targetPMS-> parameter contains the PowerManagementService responsible to
// perform the request.
```

```
    $instanceConcreteJob = smNewInstance (“CIM_ConcreteJob”);
    %InArguments[] = {newArgument(“PowerState”, #PowerState),
    newArgument(“ManagedElement”, $targetSystem->),
    newArgument(“Time”, #time)}
    %OutArguments[] = {newArgument(“Job”, $instanceConcreteJob.getObjectPath())}
    #Error = InvokeMethod ($targetPMS->,
                        “RequestPowerStateChange”,
                        %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 == #returnStatus)  {
          //The specified extrinsic method does not exist
          $OperationError = smNewInstance(“CIM_Error”);
          //CIM_ERR_METHOD_NOT_FOUND
          $OperationError.CIMStatusCode = 17;
          //Software Error
          $OperationError.ErrorType = 10;
          //Low
          $OperationError.PerceivedSeverity = 0;
          $OperationError.OwningEntity = DMTF:SMCLP;
          $OperationError.MessageID = 0x00000001;
          $OperationError.Message = “Operation is not supported”;
          &smAddError($job, $OperationError);
          &smMakeCommandStatus($job);
          }
          else
          {
             //operation failed, but no detailed error instance, need to make //one up
             //make an Error instance and associate with job for Operation
             $OperationError = smNewInstance(“CIM_Error”);
             //CIM_ERR_FAILED
             $OperationError.CIMStatusCode = 1;
             //Software Error
             $OperationError.ErrorType = 4;
             //Unknown
             $OperationError.PerceivedSeverity = 0;
             $OperationError.OwningEntity = DMTF:SMCLP;
             $OperationError.MessageID = 0x00000009;
```

```
            $OperationError.Message = “An internal software error has occurred.”;
            &smAddError($job, $OperationError);
            &smMakeCommandStatus($job);
            &smEnd;
         }
      }//if CIM op failed
      else if (0 == #returnStatus)   {
         //completed successfully
         &smCommandCompleted($job);
         &smEnd;
      }
      else if (1 == #returnStatus)   {
         //unsupported
         $OperationError = smNewInstance(“CIM_Error”);
         //CIM_ERR_NOT_SUPPORTED
         $OperationError.CIMStatusCode = 7;
         //Other
         $OperationError.ErrorType = 1;
         //Low
         $OperationError.PerceivedSeverity = 2;
         $OperationError.OwningEntity = DMTF:SMCLP;
         $OperationError.MessageID = 0x00000001;
         $OperationError.Message = “Operation is not supported.”;
         &smAddError($job, $OperationError);
         &smMakeCommandStatus($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 Mappings

The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in the *Power State Management 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_ComputerSystem

CIM_ComputerSystem is not owned by the *Power State Management Profile*. The following mappings are in addition to those stated by the profile which owns the CIM_ComputerSystem definition.

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_ComputerSystem

Command Verb	Requirement	Comments
Show	Shall	See 6.1.2.
Reset	May	See 6.1.3.
Set	May	See 6.1.4.
Start	Shall	See 6.1.5.
Stop	Shall	See 6.1.6.

6.1.1 Ordering of Results

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

- Results for CIM_ComputerSystem 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_ComputerSystem. Implementations shall support the use of the `show` verb with CIM_ComputerSystem.

The `show` verb is used to display information about the instance of CIM_ComputerSystem and the referenced properties from the associated instance of CIM_AssociatedPowerManagementService.

6.1.2.1 Show a Single Instance

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

6.1.2.1.1 Command Form

```
show <CIM_ComputerSystem single instance>
```

6.1.2.1.2 CIM Requirements

See CIM_ComputerSystem in the “CIM Elements” section of the *Power State Management Profile* for the list of mandatory properties.

6.1.2.1.3 Behavior Requirements

```
// the class definition for $instance includes two referenced properties,
// PowerState and PowerOnTime.
$instance=<CIM_ComputerSystem single instance>;
#Error=smOpReferences(
       $instance->,
       “CIM_AssociatedPowerManagementService”,
       NULL,
       NULL,
       {“PowerState”,“PowerOnTime”},
       $APMSInstancePaths[])
if (0 != #Error.code)
{
       &smProcessOpError (#Error);
       //includes &smEnd;
}
else
{
#propertynamelist[]={<array of mandatory non-key property names in CIM_ComputerSystem
   (see CIM Requirements)>}
#additionalpropertylist[]={“PowerState”,“PowerOnTime”};
$APMSinstance = $APMSInstancePaths[1];
$instance.PowerState=$APMSinstance.PowerState;
$instance.PowerOnTime=$APMSinstance.PowerOnTime;
& smShowInstancePseudoProperties(
   $instance,
   #propertynamelist[],
   #additionalpropertylist[]);
}
&smEnd;
```

6.1.2.2 Show Multiple Instances

This command form is for the show verb applied to multiple instances of CIM_ComputerSystem.

6.1.2.2.1 Command Form

```
show <CIM_ComputerSystem multiple instances>
```

6.1.2.2.2 CIM Requirements

See CIM_ComputerSystem in the “CIM Elements” section of the *Power State Management Profile* for the list of mandatory properties.

6.1.2.2.3 Behavior Requirements

6.1.2.2.3.1 Preconditions

`$containerInstance` represents the instance of CIM_AdminDomain or CIM_ComputerSystem that is associated to the targeted instances of CIM_ComputerSystem through the CIM_SystemComponent association.

6.1.2.2.3.2 Pseudo Code

```
#Error=smOpAsociators(
   $containerinstance->,
   “CIM_SystemComponent”,
   NULL,
   NULL,
   NULL,
   $CSInstancePaths[])
if (0 != #Error.code)
{
   &smProcessOpError (#Error);
   //includes &smEnd;
}
else
{
   for #i < n //n is the number of ComputerSystem instances
   {
   #Error=smOpReferences(
      $CSInstancePaths[i]->,
      “CIM_AssociatedPowerManagementService”,
      NULL,
      NULL,
      {“PowerState”,“PowerOnTime”},
      $APMSInstancePaths[])
      if (0 != #Error.code)
      {
            &smProcessOpError (#Error);
            //includes &smEnd;
      }
      else
      {
         #propertynamelist[]={<array of mandatory non-key property names in
         CIM_ComputerSystem (see CIM Requirements)>}
         #additionalpropertylist[]={“PowerState”,“PowerOnTime”};
         $APMSinstance = $APMSInstancePaths[1];
         $instance.PowerState=$APMSinstance.PowerState;
         $instance.PowerOnTime=$APMSinstance.PowerOnTime;
```

```
          &smShowInstancePseudoProperties(
             $CSInstancePaths[i],
             #propertynamelist[],
             #additionalpropertylist[]);
      }
   }
}
&smEnd;
```

6.1.3 Reset

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

6.1.3.1 Command Form

```
reset <CIM_ComputerSystem single instance>
```

6.1.3.2 CIM Requirements

```
uint32 RequestPowerStateChange(
   uint16 PowerState,
   CIM_ManagedElement REF ManagedElement,
   datetime Time,
   CIM_ConcreteJob REF Job,
   datetime TimeoutPeriod);
```

6.1.3.3 Behavior Requirements

```
#Error = &smOpAssociators(
        $instance.getObjectPath(),
        “CIM_AssociatedPowerManagementService”,
        “CIM_PowerManagementService”,
        “UserOfService”,
        “ServiceProvided”,
        NULL,
        $PMSInstancePaths[])
if (0 != #Error.code)
   {
   &smProcessOpError (#Error);
   //includes &smEnd;
   }
else if (PMSInstancePaths.length() > 0)
   {
   $PMSinstance = $PMSInstancePaths[1];
   // 5 is equivalent to Power Cycle or System Reset;
   smRequestPowerStateChange(5, instance->, NULL, PMSinstance->);
   }
```

```
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 = 0x00000001;
   $OperationError.Message = “Operation is not supported”;
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
   }
&smEnd;
```

6.1.4 Set

6.1.4.1 Set Command Form to Change the PowerState with PowerOnTime Property Value

This section describes how to implement the `set` verb that is used to set the power state of the computer system. Implementations may support the use of the `set` verb with CIM_ComputerSystem.

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 *Power State Management Profile*.

Time is entered either as a regular date-time value or as an interval value.

6.1.4.1.1 Command Form

```
set <CIM_ComputerSystem single instance> powerstate=<powerstatevalue>
       powerontime=<requestedtime>
```

6.1.4.1.2 CIM Requirements

```
uint16 CIM_AssociatedPowerManagementService.RequestedPowerState;
datetime CIM_AssociatedPowerManagementService.PowerOnTime;
uint32 RequestPowerStateChange(
   uint16 PowerState,
   CIM_ManagedElement REF ManagedElement,
   datetime Time,
   CIM_ConcreteJob REF Job,
   datetime TimeoutPeriod);
```

6.1.4.1.3 Behavior Requirements

```
// The class definition for $instance includes two referenced properties,
// PowerState and PowerOnTime.
$instance=<CIM_ComputerSystem single instance>
#propertyNames[] = {“PowerState”};
#propertyValues[] = {<powerstatevalue>};
```

```
#time = {<requestedtime>};
#Error = &smOpAssociators(
        $instance.getObjectPath(),
        “CIM_AssociatedPowerManagementService”,
        “CIM_PowerManagementService”,
        “UserOfService”,
        “ServiceProvided”,
        NULL,
        $PMSInstancePaths[])
if (0 != #Error.code)
   {
   &smProcessOpError (#Error);
   //includes &smEnd;
   }
else
   {
   $PMSinstance = $PMSInstancePaths[1];
   &smRequestPowerStateChange($PMSinstance, #propertyNames[], #time,
      #propertyValues[]);
   }
#Error=smOpReferences(
       $instance->,
       “CIM_AssociatedPowerManagementService”,
       NULL,
       NULL,
       {“PowerState”,“PowerOnTime”},
       $APMSInstancePaths[])
if (0 != #Error.code)
   {
   &smProcessOpError (#Error);
   //includes &smEnd;
   }
else
   {
   #propertynamelist[]={<array of mandatory non-key property names in
      CIM_ComputerSystem (see CIM Requirements)>}
   #additionalpropertylist[]={“PowerState”,“PowerOnTime”};
   $APMSinstance = $APMSInstancePaths[1];
   $instance.PowerState=$APMSinstance.PowerState;
   $instance.PowerOnTime=$APMSinstance.PowerOnTime;
   &smShowInstancePseudoProperties(
   $instance,
   #propertynamelist[],
   #additionalpropertylist[]);
   }
&smEnd;
```

6.1.4.2 Set Command Form to Change the PowerState with No Time Property

This section describes how to implement the `set` verb that is used to set the power state of the computer system. Implementations may support the use of the `set` verb with CIM_ComputerSystem.

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 *Power State Management Profile*.

6.1.4.2.1 Command Form

```
set <CIM_ComputerSystem single instance> powerstate=<powerstatevalue>
```

6.1.4.2.2 CIM Requirements

```
uint16 CIM_AssociatedPowerManagementService.RequestedPowerState;
uint32 RequestPowerStateChange(
   uint16 PowerState,
   CIM_ManagedElement REF ManagedElement,
   datetime Time,
   CIM_ConcreteJob REF Job,
   datetime TimeoutPeriod);
```

6.1.4.2.3 Behavior Requirements

```
// The class definition for $instance includes one referenced property,
// PowerState.
$instance=<CIM_ComputerSystem single instance>
#propertyNames[] = {“PowerState”};
#propertyValues[] = {<powerstatevalue>};
#Error = &smOpAssociators(
        $instance.getObjectPath(),
        “CIM_AssociatedPowerManagementService”,
        “CIM_PowerManagementService”,
        “UserOfService”,
        “ServiceProvided”,
        NULL,
        $PMSInstancePaths[])
if (0 != #Error.code)
   {
   &smProcessOpError (#Error);
   //includes &smEnd;
   }
else
   {
   $PMSinstance = $PMSInstancePaths[1];
   &smRequestPowerStateChange($PMSinstance, #propertyNames[], NULL,
      #propertyValues[]);
   }
   #Error=smOpReferences(
      $instance->,
      “CIM_AssociatedPowerManagementService”,
      NULL,
       NULL,
       {“PowerState”},
       $APMSInstancePaths[])
if (0 != #Error.code)
   {
   &smProcessOpError (#Error);
   //includes &smEnd;
   }
else
   {
   #propertynamelist[]={<array of mandatory non-key property names in
      CIM_ComputerSystem (see CIM Requirements)>}
   #additionalpropertylist[]={“PowerState”,};
   $APMSinstance = $APMSInstancePaths[1];
   $instance.PowerState=$APMSinstance.PowerState;
   &smShowInstancePseudoProperties(
   $instance,
   #propertynamelist[],
   #additionalpropertylist[]);
   }
&smEnd;
```

6.1.4.3 Set Command Form to Change the PowerState with #Time Value Format

This section describes how to implement the `set` verb that is used to set the power state of the computer system. Implementations may support the use of the `set` verb with CIM_ComputerSystem.

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 *Power State Management Profile*.

The property value format for #Time is defined in DSP0216.

6.1.4.3.1 Command Form

```
set <CIM_ComputerSystem single instance> powerstate=<powerstatevalue>
      powerontime#time=<timestamp>
```

6.1.4.3.2 CIM Requirements

```
uint16 CIM_AssociatedPowerManagementService.RequestedPowerState;
datetime CIM_AssociatedPowerManagementService.PowerOnTime;
uint32 RequestPowerStateChange(
   uint16 PowerState,
   CIM_ManagedElement REF ManagedElement,
   datetime Time,
   CIM_ConcreteJob REF Job,
   datetime TimeoutPeriod);
```

6.1.4.3.3 Behavior Requirements

```
// The class definition for $instance includes two referenced properties,
// PowerState and PowerOnTime.
$instance=<CIM_ComputerSystem single instance>
#propertyNames[] = {“PowerState”};
#propertyValues[] = {<powerstatevalue>};
#time = {< timestamp converted into datetime datatype timestamp format>};
#Error = &smOpAssociators(
         $instance.getObjectPath(),
         “CIM_AssociatedPowerManagementService”,
         “CIM_PowerManagementService”,
         “UserOfService”,
         “ServiceProvided”,
         NULL,
         $PMSInstancePaths[])

if (0 != #Error.code)
   {
   &smProcessOpError (#Error);
   //includes &smEnd;
   }
else
   {
   $PMSinstance = $PMSInstancePaths[1];
   &smRequestPowerStateChange($PMSinstance, #propertyNames[], #time,
      #propertyValues[]);
   }
   #Error=smOpReferences(
      $instance->,
      “CIM_AssociatedPowerManagementService”,
      NULL,
      NULL,
      {“PowerState”,“PowerOnTime”},
      $APMSInstancePaths[])
if (0 != #Error.code)
   {
   &smProcessOpError (#Error);
   //includes &smEnd;
   }
else
   {
   #propertynamelist[]={<array of mandatory non-key property names in
      CIM_ComputerSystem (see CIM Requirements)>}
   #additionalpropertylist[]={“PowerState”,“PowerOnTime”};
   $APMSinstance = $APMSInstancePaths[1];
   $instance.PowerState=$APMSinstance.PowerState;
   $instance.PowerOnTime=$APMSinstance.PowerOnTime;
   &smShowInstancePseudoProperties(
      $instance,
      #propertynamelist[],
      #additionalpropertylist[]);
   }
&smEnd;
```

6.1.4.4 Set Command Form to Change the PowerState with #Interval Value Format

This section describes how to implement the `set` verb that is used to set the power state of the computer system. Implementations may support the use of the `set` verb with CIM_ComputerSystem.

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 *Power State Management Profile*.

Time is entered in the user friendly time interval format. The property value format for #Interval is defined in DSP0216.

6.1.4.4.1 Command Form

```
set <CIM_ComputerSystem single instance> powerstate=<powerstatevalue>
      powerontime#interval=<userfriendlyinterval>
```

6.1.4.4.2 CIM Requirements

```
uint16 CIM_AssociatedPowerManagementService.RequestedPowerState;
datetime CIM_AssociatedPowerManagementService.PowerOnTime;
uint32 RequestPowerStateChange(
   uint16 PowerState,
   CIM_ManagedElement REF ManagedElement,
   datetime Time,
   CIM_ConcreteJob REF Job,
   datetime TimeoutPeriod);
```

6.1.4.4.3 Behavior Requirements

```
// the class definition for $instance includes two referenced properties,
// PowerState and PowerOnTime.
$instance=<CIM_ComputerSystem single instance>
#propertyNames[] = {“PowerState”};
#propertyValues[] = {<powerstatevalue>};
#time = {< userfriendlyinterval converted into datetime datatype interval format>};
#Error = &smOpAssociators(
        $instance.getObjectPath(),
        “CIM_AssociatedPowerManagementService”,
        “CIM_PowerManagementService”,
        “UserOfService”,
        “ServiceProvided”,
        NULL,
        $PMSInstancePaths[])
```

```
if (0 != #Error.code)
   {
   &smProcessOpError (#Error);
   //includes &smEnd;
   }
else
   {
   $PMSinstance = $PMSInstancePaths[1];
   &smRequestPowerStateChange($PMSinstance, #propertyNames[], #time,
   #propertyValues[]);
   }
   #Error=smOpReferences(
      $instance->,
      “CIM_AssociatedPowerManagementService”,
      NULL,
      NULL,
      {“PowerState”,“PowerOnTime”},
      $APMSInstancePaths[])
if (0 != #Error.code)
   {
   &smProcessOpError (#Error);
   //includes &smEnd;
   }
else
   {
   #propertynamelist[]={<array of mandatory non-key property names in
      CIM_ComputerSystem (see CIM Requirements)>}
   #additionalpropertylist[]={“PowerState”,“PowerOnTime”};
   $APMSinstance = $APMSInstancePaths[1];
   $instance.PowerState=$APMSinstance.PowerState;
   $instance.PowerOnTime=$APMSinstance.PowerOnTime;
   &smShowInstancePseudoProperties(
      $instance,
      #propertynamelist[],
      #additionalpropertylist[]);
   }
&smEnd;
```

6.1.5 Start

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

The `start` verb is used to enable a computer system.

6.1.5.1 Command Form

```
start < CIM_ComputerSystem single instance>
```

6.1.5.2 CIM Requirements

```
uint32 RequestPowerStateChange(
   uint16 PowerState,
   CIM_ManagedElement REF ManagedElement,
   datetime Time,
   CIM_ConcreteJob REF Job,
   datetime TimeoutPeriod);
```

6.1.5.3 Behavior Requirements

```
$instance=<CIM_ComputerSystem single instance>;
#Error = &smOpAssociators(
        $instance.getObjectPath(),
        “CIM_AssociatedPowerManagementService”,
        “CIM_PowerManagementService”,
        “UserOfService”,
        “ServiceProvided”,
        NULL,
        $PMSInstancePaths[])
if (0 != #Error.code)
   {
   &smProcessOpError (#Error);
   //includes &smEnd;
   }
else if (PMSInstancePaths.length() > 0)
   {
   $PMSinstance = $PMSInstancePaths[1];
   // 2 is equivalent to Power On;
   smRequestPowerStateChange(2, instance->, NULL,PMSinstance->);
   }
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 = 0x00000001;
   $OperationError.Message = “Operation is not supported”;
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
   }
&smEnd;
```

6.1.6 Stop

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

The `stop` verb is used o disable a computer system.

6.1.6.1 Command Form

```
stop < CIM_ComputerSystem single instance>
```

6.1.6.2 CIM Requirements

```
uint32 RequestPowerStateChange(
   uint16 PowerState,
   CIM_ManagedElement REF ManagedElement,
   datetime Time,
   CIM_ConcreteJob REF Job,
   datetime TimeoutPeriod);
```

6.1.6.3 Behavior Requirements

```
$instance=<CIM_ComputerSystem single instance>;
#Error = &smOpAssociators(
        $instance.getObjectPath(),
        “CIM_AssociatedPowerManagementService”,
        “CIM_PowerManagementService”,
        “UserOfService”,
        “ServiceProvided”,
        NULL,
        $PMSInstancePaths[])
if (0 != #Error.code)
   {
   &smProcessOpError (#Error);
   //includes &smEnd;
   }
else if (PMSInstancePaths.length() > 0)
   {
   $PMSinstance = $PMSInstancePaths[1];
   // 8 is equivalent to Power Off(Soft)
   smRequestPowerStateChange(8, instance->, NULL,PMSinstance->);
   }
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 = 0x00000001;
```

```
    $OperationError.Message = “Operation is not supported”;
    &smAddError($job, $OperationError);
    &smMakeCommandStatus($job);
}
&smEnd;
```

6.2 CIM_PowerManagementService

The `cd`, `help`, `version`, and `exit` 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_PowerManagementService

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`, and `load`.

6.2.1 Ordering of Results

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

- Results for CIM_PowerManagementService 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_PowerManagementService. Implementations shall support the use of the `show` verb with CIM_PowerManagementService.

The `show` verb is used to display information about CIM_PowerManagementService instances.

6.2.2.1 Show Command Form for Single Instance Target

6.2.2.1.1 Command Form

```
show <CIM_PowerManagementService single instance>
```

6.2.2.1.2 CIM Requirements

See CIM_PowerManagementService in the “CIM Elements” section of the *Power State Management Profile* for the list of mandatory properties.

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

6.2.2.2 Show Command Form for Multiple Instances Target

6.2.2.2.1 Command Form

```
show <CIM_PowerManagementService multiple instances>
```

6.2.2.2.2 CIM Requirements

See CIM_PowerManagementService in the “CIM Elements” section of the *Power State Management Profile* for the list of mandatory properties.

6.2.2.2.3 Behavior Requirements

6.2.2.2.3.1 Preconditions

`$containerInstance` contains the instance of CIM_ComputerSystem that is associated to the targeted instances of CIM_PowerManagementServices through the CIM_HostedService association.

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

6.2.2.2.3.2 Pseudo Code

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

6.3 CIM_PowerManagementCapabilities

The `cd`, `help`, `version`, and `exit` 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_PowerManagementCapabilities

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 6.3.2.
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`.

6.3.1 Ordering of Results

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

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

6.3.2 Show

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

The `show` verb is used to display information about CIM_PowerManagementCapabilities instances.

6.3.2.1 Show Command Form for a Single Instance Target

6.3.2.1.1 Command Form

```
show < CIM_PowerManagementCapabilities single instance>
```

6.3.2.1.2 CIM Requirements

See CIM_PowerManagementCapabilities in the "CIM Elements" section of the *Power State Management Profile* for the list of mandatory properties.

6.3.2.1.3 Behavior Requirements

6.3.2.1.3.1 Preconditions

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

6.3.2.1.3.2 Pseudo Code

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

6.3.2.2 Show Command Form for Multiple Instances Target

6.3.2.2.1 Command Form

```
show < CIM_PowerManagementCapabilities multiple instances>
```

6.3.2.2.2 CIM Requirements

See CIM_PowerManagementCapabilities in the "CIM Elements" section of the *Power State Management Profile* for the list of mandatory properties.

6.3.2.2.3 Behavior Requirements

6.3.2.2.3.1 Preconditions

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

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

6.3.2.2.3.2 Pseudo Code

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

6.4 CIM_AssociatedPowerManagementService

The `cd`, `help`, `version`, and `exit` 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_AssociatedPowerManagementService

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

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

6.4.1 Ordering of Results

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

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

6.4.2 Show

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

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

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

6.4.2.1.1 Command Form

```
show <CIM_AssociatedPowerManagementService single instance>
```

6.4.2.1.2 CIM Requirements

See CIM_AssociatedPowerManagementService in the “CIM Elements” section of the *Power State Management Profile* for the list of mandatory properties.

6.4.2.1.3 Behavior Requirements

6.4.2.1.3.1 Preconditions

`$instance` represents the instance of a CIM_ComputerSystem, which is referenced by CIM_AssociatedPowerManagementService.

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

6.4.2.1.3.2 Pseudo Code

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

6.4.2.2 Show Command Form for Multiple Instances Target – CIM_PowerManagementService Reference

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

6.4.2.2.1 Command Form

```
show <CIM_AssociatedPowerManagementService multiple instances>
```

6.4.2.2.2 CIM Requirements

See CIM_AssociatedPowerManagementService in the “CIM Elements” section of the *Power State Management Profile* for the list of mandatory properties.

6.4.2.2.3 Behavior Requirements

6.4.2.2.3.1 Preconditions

`$instance` represents the instance of a CIM_PowerManagementService, which is referenced by CIM_AssociatedPowerManagementService.

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

6.4.2.2.3.2 Pseudo Code

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

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

6.4.2.3 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_AssociatedPowerManagementService where both references are specified and therefore the desired instance is unambiguously identified.

6.4.2.3.1 Command Form

```
show <CIM_AssociatedPowerManagementService single instance>
```

6.4.2.3.2 CIM Requirements

See CIM_AssociatedPowerManagementService in the “CIM Elements” section of the *Power State Management Profile* for the list of mandatory properties.

6.4.2.3.3 Behavior Requirements

6.4.2.3.3.1 Preconditions

$instance represents the instanceA of a CIM_ComputerSystem, which is referenced by CIM_AssociatedPowerManagementService.

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

6.4.2.3.3.2 Pseudo Code

```
$instanceA=<CIM_ComputerSystem single instance>;
$instanceB=<CIM_PowerManagementService single instance>;
#propertylist[] = NULL;
if ( false == #all )
   {
   #propertylist[] = <array of mandatory non-key property names (see CIM
      Requirements)>;
   }
&smShowAssociationInstance ( “CIM_AssociatedPowerManagementService”,
   $instanceA.getObjectPath(), $instanceB.getObjectPath(), #propertylist[] );
&smEnd;
```

6.5 CIM_ElementCapabilities

The `cd`, `help`, `version`, and `exit` 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.
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`.

6.5.1 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_PowerManagementService Reference

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

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 *Power State Management Profile* for the list of mandatory properties.

6.5.2.1.2.1 Behavior Requirements

6.5.2.1.2.2 Preconditions

`$instance` represents the instance of a CIM_PowerManagementService, which is referenced by CIM_ElementCapabilities.

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

6.5.2.1.2.3 Pseudo Code

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

6.5.2.2 Show Command Form for Multiple Instances – CIM_PowerManagementCapabilities Reference

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

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 *Power State Management 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 a CIM_PowerManagementCapabilities, which is referenced by CIM_ElementCapabilities.

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

6.5.2.2.3.2 Pseudo Code

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

6.5.2.3 Show Command Form for Single Instance Target – CIM_PowerManagementService and CIM_PowerManagementCapabilities 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.

6.5.2.3.1 Command Form

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

6.5.2.3.2 CIM Requirements

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

6.5.2.3.3 Behavior Requirements

6.5.2.3.3.1 Preconditions

`$instanceA` represents the instance of a CIM_PowerManagementService and `$instanceB` represents the instance of a CIM_PowerManagementCapabilities, both of which are referenced by CIM_ElementCapabilities.

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

6.5.2.3.3.2 Pseudo Code

```
$instanceA=<CIM_PowerManagementService single instance>;
$instanceB=<CIM_PowerManagementCapabilities single instance>;
#propertylist[] = NULL;
if ( false == #all )
   {
   #propertylist[] = <array of mandatory non-key property names (see CIM
      Requirements)>;
   }
&smShowAssociationInstance (“CIM_ElementCapabilities”, $instanceA.getObjectPath(),
   $instanceB.getObjectPath(), #propertylist[] );
&smEnd;
```

6.6 CIM_HostedService

The `cd`, `help`, `version`, and `exit` 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_HostedService

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`, `set`, `start`, and `stop`.

6.6.1 Ordering of Results

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

- Results for CIM_HostedService 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_HostedService. Implementations shall support the use of the `show` verb with CIM_HostedService.

6.6.2.1 Show Command Form for Multiple Instances Target – CIM_ComputerSystem Reference

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

6.6.2.1.1 Command Form

```
show <CIM_HostedService multiple instances>
```

6.6.2.1.2 CIM Requirements

See CIM_HostedService in the "CIM Elements" section of the *Power State Management Profile* for the list of mandatory properties.

6.6.2.1.3 Behavior Requirements

6.6.2.1.3.1 Preconditions

`$instance` represents the instance of a CIM_ComputerSystem, which is referenced by CIM_HostedService.

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

6.6.2.1.3.2 Pseudo Code

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

6.6.2.2 Show Command Form for a Single Instance – CIM_PowerManagementService Reference

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

6.6.2.2.1 Command Form

```
show <CIM_HostedService single instance>
```

6.6.2.2.2 CIM Requirements

See CIM_HostedService in the “CIM Elements” section of the *Power State Management 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 instance of a CIM_PowerManagementService, which is referenced by CIM_HostedService.

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

6.6.2.2.3.2 Pseudo Code

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

6.6.2.3 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_HostedService where both references are specified and therefore the desired instance is unambiguously identified.

6.6.2.3.1 Command Form

```
show <CIM_HostedService single instance>
```

6.6.2.3.2 CIM Requirements

See CIM_HostedService in the “CIM Elements” section of the *Power State Management Profile* for the list of mandatory properties.

6.6.2.3.3 Behavior Requirements

6.6.2.3.3.1 Preconditions

`$instanceA` represents the instance of a CIM_ComputerSystem and `$instanceB` represents the instance of CIM_PowerManagementservice, both of which are referenced by CIM_HostedService.

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

6.6.2.3.3.2 Pseudo Code

```
$instanceA=<CIM_ComputerSystem single instance>;
$instanceB=<CIM_PowerManagementService single instance>;
#propertylist[] = NULL;
if ( false == #all )
   {
   #propertylist[] = <array of mandatory non-key property names (see CIM
      Requirements)>;
   }
&smShowAssociationInstance ( “CIM_HostedService”, $instanceA.getObjectPath(),
   $instanceB.getObjectPath(), #propertylist[]);
&smEnd;
```

ANNEX A
(informative)

Change Log

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