Document Number: DSP0813

Date: 2009-06-04

Version: 1.0.0

Boot Control Profile SM CLP Command Mapping Specification

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

CONTENTS

Tables

75

Foreword

The *Boot Control Profile SM CLP Command Mapping Specification* (DSP0813) was prepared by the Server Management Working Group.

Conventions

The pseudo code conventions utilized in this document are the Recipe Conventions as defined in the 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:

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

Introduction

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

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

Boot Control 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 *Boot Control 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 DSP1012, *Boot Control Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1012_1.0.pdf

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

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

2.2 Other References

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

3 Terms and Definitions

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

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

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

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

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

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

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

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

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

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

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

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

4 Symbols and Abbreviated Terms

The following symbols and abbreviations are used in this document.

4.1
CIM
Common Information Model

4.2
CLP
Command Line Protocol

4.3
DMTF
Distributed Management Task Force

4.4
IETF
Internet Engineering Task Force

4.5
SM
Server Management

4.6
SMI-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 *SM CLP-to-CIM Common Mapping Specification 1.0* (DSP0216).

- smOpDeleteInstance()
- smReset()
- smShowInstance()
- smShowInstances()
- smSetInstance()
- smShowAssociationInstances()

This mapping does not define any recipes for local reuse.

6 Mappings

The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in the *Boot Control 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_BootService

The `cd` and `help` verbs shall be supported as described in DSDP0216.

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.

208 **Table 1 – Command Verb Requirements for CIM_BootService**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	May	See 6.1.2.
show	Shall	See 6.1.3.
start	Not supported	
stop	Not supported	

209 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `exit`,
210 `load`, `reset`, `start`, and `stop`.

211 ### 6.1.1 Ordering of Results

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

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

215 ### 6.1.2 Set

216 The `set` verb is used to set properties on an instance of CIM_BootService. Implementations may support
217 the use of the `set` verb with CIM_BootService.

218 #### 6.1.2.1 General Usage of Set for a Single Property

219 This command form corresponds to the general usage of the set verb to modify a single property of a
220 target instance.

221 The requirements for supporting modification of a property using this command form shall be equivalent
222 to the requirement for supporting modification of the property using the ModifyInstance operation as
223 defined in the *Boot Control Profile*.

224 ##### 6.1.2.1.1 Command Form

225 **`set <CIM_BootService *single instance*> <propertyname>=<propertyvalue>`**

226 ##### 6.1.2.1.2 CIM Requirements

227 See CIM_BootService in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory
228 properties.

6.1.2.1.3 Behavior Requirements

6.1.2.1.3.1 Pseudo Code

```
$instance=<CIM_BootService single instance>
#propertyNames[] = <propertname>
#propertyValues[] = <propertyvalue>
&smSetInstance ( $instance, #propertyNames, #propertyValues );
&smEnd;
```

6.1.2.2 General Usage of Set for Multiple Properties

This command form corresponds to the general usage of the `set` verb to modify a multiple properties of a target instance where there isn't an explicit relationship between the properties.

The requirements 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 *Boot Control Profile*.

6.1.2.2.1 Command Form

```
set <CIM_BootService single instance> <propertyname1>=<propertyvalue1>
   <propertynameN>=<propertyvalueN>
```

6.1.2.2.2 CIM Requirements

See CIM_BootService in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.1.2.2.3 Behavior Requirements

6.1.2.2.3.1 Preconditions

`$instance` represents the instance of CIM_BootService.

6.1.2.2.3.2 Pseudo Code

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

6.1.3 Show

The `show` verb is used to display information about instances of CIM_BootService. Implementations shall support the use of the `show` verb with CIM_BootService.

6.1.3.1 Show a Single Instance

This command form is used to display information about a single instance of CIM_BootService.

6.1.3.1.1 Command Form

```
show <CIM_BootService single instance>
```

6.1.3.1.2 CIM Requirements

See CIM_BootService in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.1.3.1.3 Behavior Requirements

6.1.3.1.3.1 Preconditions

`$instance` represents the instance of CIM_BootService.

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

`#propertylist[]` is an array of mandatory non-key property names.

6.1.3.1.3.2 Pseudo Code

```
if (false != #all) { #propertylist[] = NULL; }
&smShowInstance ( $instance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.1.3.2 Show Multiple Instances

This command form is used to display information about multiple instances of CIM_BootService.

6.1.3.2.1 Command Form

```
show <CIM_BootService multiple instances>
```

6.1.3.2.2 CIM Requirements

See CIM_BootService in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.1.3.2.3 Behavior Requirements

6.1.3.2.3.1 Preconditions

`$containerInstance` represents the instance of CIM_ComputerSystem to which the instances of CIM_BootService are scoped. The CIM_BootService is associated to CIM_ComputerSystem via a CIM_HostedService association.

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

`#propertylist[]` is an array of mandatory non-key property names.

6.1.3.2.3.2 Pseudo Code

```
if (false != #all) { #propertylist[] = NULL; }
&smShowInstances ( “CIM_BootService”, “CIM_HostedService”,
   $containerInstance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.2 CIM_BootServiceCapabilities

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 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_BootServiceCapabilities

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 mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `exit`, `load`, `reset`, `set`, `start`, and `stop`.

6.2.1 Ordering of Results

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

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

6.2.2 Show

The `show` verb is used to display information about instances of CIM_BootServiceCapabilities. Implementations shall support the use of the `show` verb with CIM_BootServiceCapabilities.

6.2.2.1 Show a Single Instance

This command form is used to display the information about a single instance of CIM_BootServiceCapabilities.

6.2.2.1.1 Command Form

```
show <CIM_BootServiceCapabilities single instance>
```

6.2.2.1.2 CIM Requirements

See CIM_BootServiceCapabilities in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.2.2.1.3 Behavior Requirements

6.2.2.1.3.1 Preconditions

`$instance` represents the instance of CIM_BootServiceCapabilities.

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

`#propertylist[]` is an array of mandatory non-key property names.

6.2.2.1.3.2 Pseudo Code

```
if (false != #all) { #propertylist[] = NULL; }
&smShowInstance ( $instance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.2.2.2 Show Multiple Instances

This command form is used to display the information about multiple instances of CIM_BootServiceCapabilities. This command form corresponds to UFsT-based selection within a scoping system.

6.2.2.2.1 Command Form

```
show <CIM_BootServiceCapabilities multiple instances>
```

6.2.2.2.2 CIM Requirements

See CIM_BootServiceCapabilities in the "CIM Elements" section of the *Boot Control Profile* for the list of mandatory properties.

6.2.2.2.3 Behavior Requirements

6.2.2.2.3.1 Preconditions

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

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

`#propertylist[]` is an array of mandatory non-key property names.

6.2.2.2.3.2 Pseudo Code

```
if (false != #all) { #propertylist[] = NULL; }
&smShowInstances ( "CIM_BootServiceCapabilities", "CIM_MemberOfCollection",
   $containerInstance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.3 CIM_BootConfigSetting

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_BootConfigSetting

Command Verb	Requirement	Comments
create	May	See 6.3.2.
delete	May	See 6.3.3.
dump	Not supported	
load	Not supported	
reset	Not supported	
set	May	See 6.3.4.
show	Shall	See 6.3.5.
start	Not supported	
stop	Not supported	

No mappings are defined for the following verbs for the specified target: `dump`, `exit`, `load`, `reset`, `start`, and `stop`.

6.3.1 Ordering of Results

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

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

6.3.2 Create

The `create` verb is used to create an instance of CIM_BootConfigSetting and associated instances using a template CIM_BootConfigSetting instance. Implementations may support the use of the `create` verb with CIM_BootConfigSetting.

The template CIM_BootConfigSetting instance can be specified in the command. Otherwise, the template CIM_BootConfigSetting instance shall be the CIM_BootConfigSetting instance whose CIM_ElementSettingData.IsCurrent property has the value of 1 (IsDefault).

6.3.2.1 Create a Single Instance from a Specified Boot Configuration

This command form is used to create an instance of CIM_BootConfigSetting using an existing CIM_BootConfigSetting as the template.

6.3.2.1.1 Command Form

```
create –source <BootConfigTemplatePath> <CIM_BootConfigSetting single instance>
```

6.3.2.1.2 CIM Requirements

See CIM_BootConfigSetting in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.3.2.1.3 Behavior Requirements

6.3.2.1.3.1 Preconditions

`$instanceTemplatePath` represents a path to an instance of CIM_BootConfigSetting to use as the creation template. `$instanceTemplatePath` is translated from the <BootConfigTemplatePath>.

385 $computerInstance represents the instance of CIM_ComputerSystem which is associated to Setting
386 Collection via a CIM_OwningCollectionElement association. The Setting Collection is the instance of
387 CIM_ConcreteCollection whose ElementName property is set to "Settings" which can be translated from
388 the <CIM_BootConfigSetting single instance> path.

389 #propertylist[]is an array of mandatory non-key property names.

390 **6.3.2.1.3.2 Pseudo Code**

```
391 //Step 1: Verify Template Boot Configuration exists
392 #Error = &smOpGetInstance ($instanceTemplatePath->, NULL, $instanceTemplate);
393 if (0 != Error.code)
394    {
395    &smProcessOpError (#Error);
396    }
397 // Step 2: Try to find the CIM_ComputerSystem
398 #Error = smOpAssociators($instanceTemplatePath.getObjectPath(),
399    “CIM_ElementSettingData”, “CIM_ComputerSystem”, “ManagedElement”, “SettingData”,
400    NULL, $System->[]);
401 if (0 != Error.code)
402    {
403    &smProcessOpError (#Error);
404    }
405 //no system associated
406 if (null == $System[0])
407    {
408    $OperationError = smNewInstance(“CIM_Error”);
409    //CIM_ERR_FAILED
410    $OperationError.CIMStatusCode = 1;
411    //Software Error
412    $OperationError.ErrorType = 4;
413    //Unknown
414    $OperationError.PerceivedSeverity = 0;
415    $OperationError.OwningEntity = DMTF:SMCLP;
416    $OperationError.MessageID = 0x00000001;
417    $OperationError.Message = “Operation is not supported”;
418    &smAddError($job, $OperationError);
419    &smMakeCommandStatus($job);
420    &smEnd;
421 }
422 // Step 2: Try to find the CIM_BootService
423 #Error = smOpAssociators($System[0].getObjectPath(), “CIM_ServiceAffectsElement”,
424    “CIM_BootService”, “AffectingElement”, “AffectedElement”, NULL, Services->[]);
425 if (0 != Error.code)
426    {
427    &smProcessOpError (#Error);
428    }
429 //no service associated
430 if (null == Services[0])
431    {
432    $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 = 0x00000001;
   $OperationError.Message = “Operation is not supported”;
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
   &smEnd;
   }
//Take the first instance we find
$Service-> = $Services[0].getObjectPath();
//Step 2: Build parameter lists for method invocation
%InArguments[] =
   {
   newArgument(“ScopingComputerSystem”, $computerInstance)
   newArgument(“StartBootConfig”, $instanceTemplatePath.getObjectPath())
   };
%OutArguments[] =
   {
   newArgument(“Job”, instanceConcreteJob.getObjectPath())
   newArgument(“NewBootConfig”, $instanceBootConfigSetting)
   };
//Step 3: Invoke method
#Error = smOpInvokeMethod ($Service->, “CreateBootConfig”, %InArguments[],
   %OutArguments[], returnStatus);
//Step 4: Process return code to CLP Command Status
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;
```

```
      $OperationError.Message = “Operation is not supported.”
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   } else {
      //operation failed, but no detailed error instance, need to make one
      //up make an Error instance and associate with job for Operation
      $OperationError = smNewInstance(“CIM_Error”);
      //CIM_ERR_FAILED
      $OperationError.CIMStatusCode = 1;
      //Software Error
      $OperationError.ErrorType = 4;
      //Unknown
      $OperationError.PerceivedSeverity = 0;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000009;
      $OperationError.Message = “An internal software error has occurred.”;
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   }
} else if (0 == #returnStatus)   {
   // Method completed successfully, show new boot configuration
   &smShowInstance($instance.getObjectPath(), #propertylist[] );
   &smCommandCompleted($job);
   &smEnd;
} else if (4096 == #returnStatus) {
   //job spawned, need to watch for it to finish
   //while the jobstate is 4 (“Running”)
   while (4 == $instanceConcreteJob.JobState){<busy wait>}
   //when job finishes, invoke GetError()
   if (2 != $job.OperationalStatus) {
      %InArguments[] = { }
      %OutArguments[] = {newArgument(“Job”, $instanceConcreteJob.getObjectPath())}
      #Error = smOpInvokeMethod($job, “GetError”, %InArguments, %OutArguments,
         #returncode);

      //Method invocation failed, internal processing error
      if ( (0 != #Error.code) || (0 != #returncode) )  {
         //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;
        } else {
            //make command status
            $joberror = %OutArguments[“Error”];
            &smMakeCommandExecutionFailed($job, {$joberror};
        } //end if have CIM_Error from GetError()
    }
} 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;
}
```

6.3.2.2 Create a Single Instance from the Default Boot Configuration

This command form is used to create an instance of CIM_BootConfigSetting using an existing CIM_BootConfigSetting with the role of "Is Default".

6.3.2.2.1 Command Form

```
create <CIM_BootConfigSetting single instance>
```

6.3.2.2.2 CIM Requirements

See CIM_BootConfigSetting in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.3.2.2.3 Behavior Requirements

6.3.2.2.3.1 Preconditions

`$computerInstance` represents the instance of CIM_ComputerSystem which is associated to Setting Collection via a CIM_OwningCollectionElement association. The Setting Collection is the instance of CIM_ConcreteCollection whose ElementName property is set to "Settings" which can be translated from <CIM_BootConfigSetting single instance> path.

`#propertylist[]`is an array of mandatory non-key property names.

6.3.2.2.4 Pseudo Code

```
// Step 1: Try to find the CIM_BootService
   #Error = smOpAssociators($computerInstance.getObjectPath(),
      “CIM_ServiceAffectsElement”, “CIM_BootService”, “AffectingElement”,
      “AffectedElement”, NULL, Services->[]);
if (0 != Error.code)
   {
   &smProcessOpError (#Error);
   }
//no service associated
if (null == Services[0])
   {
   $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 = 0x00000001;
   $OperationError.Message = “Operation is not supported.”;
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
   &smEnd;
   }
//Take the first instance we find
$Service-> = $Services[0].getObjectPath();
//Step 2: Build parameter lists for method invocation
%InArguments[] = {
   newArgument(“ScopingComputerSystem”, $computerInstance)
   newArgument(“StartBootConfig”, NULL)
};
%OutArguments[] = {
   newArgument(“Job”, instanceConcreteJob.getObjectPath())
   newArgument(“NewBootConfig”, $instanceBootConfigSetting)
};
// Step 3: Invoke method
#Error = smOpInvokeMethod ($Service->, “CreateBootConfig”, %InArguments[],
   %OutArguments[], returnStatus);
// Step 4: Process return code to CLP Command Status
if (0 != #Error.code) {
   //method invocation failed
   if ( (null != #Error.$error) && (null != #Error.$error[0]) )    {
      //if the method invocation contains an embedded error
      //use it for the Error for the overall job
      &smAddError($job, #Error.$error[0]);
      &smMakeCommandStatus($job);
      &smEnd;
```

```
   } else if ( 17 == #Error.code ) {
      //17 – CIM_ERR_METHOD_NOT_FOUND
      // The specified extrinsic method does not exist.
      $OperationError = smNewInstance(“CIM_Error”);
      // CIM_ERR_METHOD_NOT_FOUND
      $OperationError.CIMStatusCode = 17;
      //Software Error
      $OperationError.ErrorType = 10;
      //Unknown
      $OperationError.PerceivedSeverity = 0;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000001;
      $OperationError.Message = “Operation is not supported.”
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   } else {
      //operation failed, but no detailed error instance, need to make one
      //up make an Error instance and associate with job for Operation
      $OperationError = smNewInstance(“CIM_Error”);
      //CIM_ERR_FAILED
      $OperationError.CIMStatusCode = 1;
      //Software Error
      $OperationError.ErrorType = 4;
      //Unknown
      $OperationError.PerceivedSeverity = 0;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000009;
      $OperationError.Message = “An internal software error has occurred.”;
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   }
} else if (0 == #returnStatus) {
   // Method completed successfully, show new boot configuration
   &smDisplayInstance ($instanceBootConfigSetting);
   &smCommandCompleted($job);
   &smEnd;
} else if (4096 == #returnStatus) {
   //job spawned, need to watch for it to finish
   //while the jobstate is 4 (“Running”)
   while (4 == $instanceConcreteJob.JobState){<busy wait>}
   //when job finishes, invoke GetError()
   if (2 != $job.OperationalStatus) {
      %InArguments[] = { }
      %OutArguments[] = {newArgument(“Job”, $instanceConcreteJob.getObjectPath())}
      #Error = smOpInvokeMethod($job, “GetError”, %InArguments, %OutArguments,
         #returncode);
      //Method invocation failed, internal processing error
```

```
      if ( (0 != #Error.code) || (0 != #returncode) )  {
         //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;
      } else {
         //make command status
         $joberror = %OutArguments["Error"];
         &smMakeCommandExecutionFailed($job, {$joberror};
      } //end if have CIM_Error from GetError()
   } //embedded job not OK
} 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 Delete

The `delete` verb is used to delete instances CIM_BootConfigSetting.

Implementations may support the use of the `delete` verb with CIM_BootConfigSetting.

6.3.3.1 Delete a Single Instance

This command form is used to delete a single instance of CIM_BootConfigSetting.

6.3.3.1.1 Command Form

```
delete <CIM_BootConfigSetting single instance>
```

6.3.3.1.2 CIM Requirements

See CIM_BootConfigSetting in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.3.3.1.3 Behavior Requirements

6.3.3.1.3.1 Preconditions

`$instance` represents the instance of CIM_BootConfigSetting.

6.3.3.1.3.2 Pseudo Code

```
#Error = &smOpDeleteInstance ($instance);
if (0 != Error.code)
   {
   &smProcessOpError (#Error);
   }
&smEnd;
```

6.3.4 Set

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

6.3.4.1 Set the BootOrder Referenced Property

This command form is used to change the boot order within a Boot Configuration. The *bootorder* referenced property is the mechanism of providing the boot sequence to the command. On the command line, the *bootorder* referenced property is set to a list of CIM_BootSourceSetting.ElementName properties. The order of the list is interpreted as the boot sequence. Each item on the list is verified to match a CIM_BootSourceSetting associated to the target CIM_BootConfigSetting prior to calling the ChangeBootOrder( ) method.

6.3.4.1.1 Command Form

`set <CIM_BootConfigSetting` *`single instance`*`> bootorder=<BootSource1>,…,<BootSourceN>`

6.3.4.1.2 CIM Requirements

See CIM_BootConfigSetting in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.3.4.1.3 Behavior Requirements

6.3.4.1.3.1 Preconditions

`$instance` represents the instance of CIM_BootConfigSetting.

`$bootSources[]` is an array of BootSource instances from the command line.

`#propertylist2[]` is an array of mandatory non-key property names for CIM_OrderedComponent.

`#propertylist3[]` is an array of mandatory non-key property names for CIM_BootSourceSetting.

6.3.4.1.3.2 Pseudo Code

```
// Step 1: Get the CIM_BootConfigSetting associated with each of
// CIM_BootSourceSetting in the boot order. Verify that
// CIM_BootConfigSetting for each boot source is the
// target CIM_BootConfigSetting.
// If one of the names is not found abort the command. Otherwise,
// save the object path in $bootOrder->[] array.
for #i in $bootSources[] {
   #Error = &smOpAssociators( $bootSource[#i], “CIM_OrderedComponent”,
      “CIM_BootConfigSetting”, NULL, NULL, NULL, $bootConfig[] );
   if (0 != #Error.code) {
      &smProcessOpError (#Error);
      //includes &smEnd;
   }
   if ( $bootConfig[0] == $instance ) {
      $bootOrder->[#i] = $bootSources[#j]->;
      continue;
   } else {
      if ($instanceDefaultBootConfig == NULL) {
         $OperationError = smNewInstance(“CIM_Error”);
         $OperationError.CIMStatusCode = 1;
         $OperationError.ErrorType = 4;
         $OperationError.PerceivedSeverity = 0;
         $OperationError.OwningEntity = DMFT:SMCLP;
         $OperationError.MessageID = 0x00000009;
         $OperationError.Message = “An internal software error has occurred.”;
         &smAddError($job, $OperationError);
         &smMakeCommandStatus($job);
         &smEnd;
      }
   }
}
//Step 2: Build parameter lists for method invocation
%InArguments[] = {newArgument(“source”, $bootOrder->[])}
%OutArguments[] = {
   newArgument(“Job”, instanceConcreteJob.getObjectPath())
}
//step 3, invoke method
#Error = smOpInvokeMethod ($instance->,
   “ChangeBootOrder”, %InArguments[], %OutArguments[], #returnStatus);
//step 4, process return code to CLP Command Status
if (0 != #Error.code)  {
   //method invocation failed
   if ( (null != #Error.$error) && (null != #Error.$error[0]) )
   {
      //if the method invocation contains an embedded error
      //use it for the Error for the overall job
      &smAddError($job, #Error.$error[0]);
      &smMakeCommandStatus($job);
```

```
      &smEnd;
   }
   else if ( 17 == #Error.code )
   {
      //17 – CIM_ERR_METHOD_NOT_FOUND
      // The specified extrinsic method does not exist.
      $OperationError = smNewInstance(“CIM_Error”);
      // CIM_ERR_METHOD_NOT_FOUND
      $OperationError.CIMStatusCode = 17;
      //Software Error
      $OperationError.ErrorType = 10;
      //Unknown
      $OperationError.PerceivedSeverity = 0;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000001;
      $OperationError.Message = “Operation is not supported.”
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else
   {
      //operation failed, but no detailed error instance, need to make one
      //up make an Error instance and associate with job for Operation
      $OperationError = smNewInstance(“CIM_Error”);
      //CIM_ERR_FAILED
      $OperationError.CIMStatusCode = 1;
      //Software Error
      $OperationError.ErrorType = 4;
      //Unknown
      $OperationError.PerceivedSeverity = 0;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000009;
      $OperationError.Message = “An internal software error has occurred.”;
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   }
}//if CIM op failed
else if (0 == #returnStatus)
   {
   //completed successfully
   // Show boot sources in order
   &smOpReferences($instance.getObjectPath(), “CIM_OrderedComponent”, NULL, NULL,
      NULL, $associatedBootOrder->[]);
   &smSortInstancePaths($associatedBootOrder->[], “AssignedSequence”,
      “AscendingOrder”, $orderedBootOrder->[]);
   for #i in $orderedBootOrder[]
   {
```

```
847         $smDisplayInstance($orderedBootOrder->[#i], propertylist2[]);
848         &smOpGetInstance($orderedBootOrder->[#i].PartComponent, NULL, $bootSource);
849         $smShowInstance($bootSource.GetObjectPath, propertylist3[]);
850      }
851      &smCommandCompleted($job);
852      &smEnd;
853   }
854   else if (4096 == #returnStatus)  {
855      //job spawned, need to watch for it to finish
856      //while the jobstate is 4 (“Running”)
857      while (4 == $instanceConcreteJob.JobState){<busy wait>}
858      //when job finishes, invoke GetError()
859      if (2 != $job.OperationalStatus) {
860         %InArguments[] = { }
861         %OutArguments[] = {newArgument(“Job”, $instanceConcreteJob.getObjectPath())}
862         #Error = smOpInvokeMethod($job,
863                   “GetError”
864                   %InArguments,
865                   %OutArguments,
866                   #returncode);
867         //Method invocation failed, internal processing error
868         if ( (0 != #Error.code) || (0 != #returncode) )  {
869         //make an Error instance and associate with job for Operation
870            $OperationError = smNewInstance(“CIM_Error”);
871            //CIM_ERR_FAILED
872            $OperationError.CIMStatusCode = 1;
873            //Software Error
874            $OperationError.ErrorType = 4;
875            //Unknown
876            $OperationError.PerceivedSeverity = 0;
877            $OperationError.OwningEntity = DMTF:SMCLP;
878            $OperationError.MessageID = 0x00000009;
879            $OperationError.Message = “An internal software error has occurred.”;
880            &smAddError($job, $OperationError);
881            &smMakeCommandStatus($job);
882            &smEnd;
883         } else    {
884            //make command status
885            $joberror = %OutArguments[“Error”];
886            &smMakeCommandExecutionFailed($job, {$joberror};
887         }//end if have CIM_Error from GetError()
888      }//embedded job not OK
889   } else {
890      //unspecified return code, generic failure.
891      $OperationError = smNewInstance(“CIM_Error”);
892      //CIM_ERR_FAILED
893      $OperationError.CIMStatusCode = 1;
894      //Other
895      $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;
}
```

6.3.5 Show

The `show` verb is used to display information about instances of CIM_BootConfigSetting. Implementations shall support the use of the `show` verb with CIM_BootConfigSetting.

6.3.5.1 Show a Single Instance

This command form is used to display the information about a single instance of CIM_BootConfigSetting.

6.3.5.1.1 Command Form

```
show <CIM_BootConfigSetting single instance>
```

6.3.5.1.2 CIM Requirements

See CIM_BootConfigSetting in the "CIM Elements" section of the *Boot Control Profile* for the list of mandatory properties.

6.3.5.1.3 Behavior Requirements

6.3.5.1.3.1 Preconditions

`$instance` represents the instance of CIM_BootSettingData.

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

`#propertylist1[]` is an array of mandatory non-key property names for CIM_BootConfigSetting.

`#propertylist2[]` is an array of mandatory non-key property names for CIM_OrderedComponent.

`#propertylist3[]` is an array of mandatory non-key property names for CIM_BootSourceSetting.

6.3.5.1.3.2 Pseudo Code

```
if (false != #all) { #propertylist1[] = NULL; }
&smShowInstance ( $instance.getObjectPath(), #propertylist1[] );
// Show boot sources in order
&smOpReferences ( $instance.getObjectPath(), "CIM_OrderedComponent", NULL, NULL, NULL,
    $associatedBootOrder->[] );
&smSortInstancePaths ( $associatedBootOrder->[], "AssignedSequence", "AscendingOrder",
    $orderedBootOrder->[] );
for #i in $orderedBootOrder[]
    {
    $smDisplayInstance ( $orderedBootOrder->[#i], #propertylist2[] );
    &smOpGetInstance ( $orderedBootOrder->[#i].PartComponent, NULL, $bootSource );
    $smShowInstance ( $bootSource, #propertylist3[] );
    }
&smEnd;
```

6.3.5.2 Show Multiple Instances

This command form applies the show verb to multiple instances of CIM_BootConfigSetting. This command form uses the UFsT as the target.

6.3.5.2.1 Command Form

```
show <CIM_BootConfigSetting multiple instances>
```

6.3.5.2.2 CIM Requirements

See CIM_BootConfigSetting in the "CIM Elements" section of the *Boot Control Profile* for the list of mandatory properties.

6.3.5.2.3 Behavior Requirements

6.3.5.2.3.1 Preconditions

`$containerInstance` represents the instance of CIM_ComputerSystem for which the CIM_BootConfigSetting instances are being displayed. The CIM_BootConfigSetting instances are associated to CIM_ComputerSystem via an instance of the CIM_ElementSettingData association.

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

`#propertylist[]` is an array of mandatory non-key property names.

6.3.5.2.3.2 Pseudo Code

```
if (false != #all) { #propertylist[] = NULL; }
&smShowInstances ( "CIM_BootConfigSetting", "CIM_ElementSettingData",
   $containerInstance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.4 CIM_BootSettingData

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_BootSettingData

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	

Command Verb	Requirement	Comments
show	Shall	See 6.4.2.
start	Not supported	
stop	Not supported	

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

6.4.1 Ordering of Results

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

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

6.4.2 Show

The `show` verb is used to display information about instances of CIM_BootSettingData. Implementations shall support the use of the `show` verb with CIM_BootSettingData.

6.4.2.1 Show a Single Instance

This command form is used to display the information about a single instance of CIM_BootSettingData.

6.4.2.1.1 Command Form

```
show <CIM_BootSettingData single instance>
```

6.4.2.1.2 CIM Requirements

See CIM_BootSettingData in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.4.2.1.3 Behavior Requirements

6.4.2.1.3.1 Preconditions

In this section, `$instance` represents the instance of CIM_BootSettingData.

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

`#propertylist[]`is an array of mandatory non-key property names.

6.4.2.1.3.2 Pseudo Code

```
if (false != #all) { #propertylist[] = NULL; }
&smShowInstance ( $instance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.4.2.2 Show Multiple Instances

This command form is used to display the information about multiple instances of CIM_BootConfigSetting. This command form corresponds to UFsT-based selection within a scoping system.

6.4.2.2.1 Command Form

```
show <CIM_BootSettingData multiple instances>
```

6.4.2.2.2 CIM Requirements

See CIM_BootSettingData in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.4.2.2.3 Behavior Requirements

6.4.2.2.3.1 Preconditions

In this section, `$containerInstance` represents the instance of CIM_BootConfigSetting or CIM_BootSourceSetting that contains the instance of CIM_BootSettingData.

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

`#propertylist[]`is an array of mandatory non-key property names.

6.4.2.2.3.2 Pseudo Code

```
if (false != #all) { #propertylist[] = NULL; }
&smShowInstances ( “CIM_BootSettingData”, “CIM_ConcreteComponent”,
   $containerInstance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.5 CIM_BootSourceSetting

The `cd` and `help` 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_BootSourceSetting

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 mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `exit`, `load`, `reset`, `start`, and `stop`.

6.5.1 Ordering of Results

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

- Results for CIM_BootSourceSetting are ordered based on the AssignedSequence property of the CIM_OrderedComponent used to associate the instance of CIM_BootSourceSetting with the instance of CIM_BootConfigSetting.
- The order of display will be in increasing values of CIM_OrderedComponent.AssignedSequence property.
- When the "-all" option is present on the command line, the instances of CIM_BootSourceSetting whose associated CIM_OrderedComponent.AssignedSequence property matches 0 (zero) shall be displayed. These instances of CIM_BootSourceSetting will be displayed following the CIM_BootSourceSetting instances with non-zero AssignedSequence values. These instances of CIM_BootSourceSetting are unordered and no algorithm is defined.

6.5.2 Show

The `show` verb is used to display information about instances of CIM_BootSourceSetting. Implementations shall support the use of the `show` verb with CIM_BootSourceSetting.

6.5.2.1 Show a Single Instance

This command form is used to display the information about a single instance of CIM_BootSourceSetting.

6.5.2.1.1 Command Form

```
show <CIM_BootSourceSetting single instance>
```

6.5.2.1.2 CIM Requirements

See CIM_BootSourceSetting in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.5.2.1.3 Behavior Requirements

6.5.2.1.3.1 Preconditions

`$instance` represents the instance of CIM_BootSourceSetting.

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

`#propertylist[]`is an array of mandatory non-key property names.

6.5.2.1.3.2 Pseudo Code

```
if (false != #all) { #propertylist[] = NULL; }
&smShowInstance ( $instance.getObjectPath(), #propertylist[] );
// If the boot source is a logical identity to CIM_BootConfigSetting,
// show the association.
&smOpReferences ( $instance.getObjectPath(), “CIM_LogicalIdentity”, “SystemElement”,
   NULL, NULL, $reference[] );
&smShowInstance ( $reference->[0]), NULL );
&smEnd;
```

6.5.2.2 Show Multiple Instances

This command form is used to display the information about multiple instances of CIM_BootSourceSetting. This command form uses the UFsT as the target.

6.5.2.2.1 Command Form

```
show <CIM_BootSourceSetting multiple instances>
```

6.5.2.2.2 CIM Requirements

See CIM_BootSourceSetting in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.5.2.2.3 Behavior Requirements

6.5.2.2.3.1 Preconditions

`$containerInstance` represents the instance of CIM_BootConfigSetting for which the CIM_BootSourceSetting instances are being displayed. The CIM_BootSourceSetting instances are associated to CIM_BootConfigSetting via an instance of the CIM_OrderedComponent association.

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

`#propertylist[]`is an array of mandatory non-key property names.

6.5.2.2.3.2 Pseudo Code

```
if (false != #all) { #propertylist[] = NULL; }
&smOpReferences ( $containerInstance, “CIM_OrderedComponent”, NULL, NULL, NULL,
   $associatedBootOrder->[] );
&smSortInstancePaths ( $associatedBootOrder->[], “AssignedSequence”, “AscendingOrder”,
   $orderedBootOrder->[] );
for #i in $orderedBootOrder[]
{
   if ( $orderedBootOrder[#i].AssignedSequence != 0 )
   {
      $smDisplayInstance ( $orderedBootOrder->[#i], propertylist2[] );
      &smOpGetInstance ( $orderedBootOrder->[#i].PartComponent, NULL, $bootSource );
      $smShowInstance ( $bootSource.GetObjectPath, propertylist3[] );
   }
}
&smEnd;
```

6.6 CIM_ConcreteComponent

The `cd` and `help` 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.

1095 **Table 6 – 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 6.6.2.
Start	Not supported	
Stop	Not supported	

1096 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `exit`,
1097 `load`, `reset`, `set`, `start`, and `stop`.

1098 ### 6.6.1 Ordering of Results

1099 When results are returned for multiple instances of CIM_ConcreteComponent, implementations shall
1100 utilize the following algorithm to produce the natural (that is, default) ordering:

1101 - Results for CIM_ConcreteComponent are unordered; therefore, no algorithm is defined.

1102 ### 6.6.2 Show

1103 This section describes how to implement the `show` verb when applied to an instance of
1104 CIM_ConcreteComponent. Implementations shall support the use of the `show` verb with
1105 CIM_ConcreteComponent.

1106 #### 6.6.2.1 Show Multiple Instances – CIM_BootConfigSetting Reference

1107 This command form is used when the `show` verb applies to multiple instances. This command form
1108 corresponds to a `show` command issued against instances of CIM_ConcreteComponent where only one
1109 reference is specified and the reference is to the instance of CIM_BootConfigSetting.

1110 ##### 6.6.2.1.1 Command Form

1111 **`show <CIM_ConcreteComponent multiple instances>`**

1112 ##### 6.6.2.1.2 CIM Requirements

1113 See CIM_ConcreteComponent in the “CIM Elements” section of the *Boot Control Profile* for the list of
1114 mandatory properties.

1115 ##### 6.6.2.1.3 Behavior Requirements

1116 ###### 6.6.2.1.3.1 Preconditions

1117 `$instance` represents the instance of CIM_BootConfigSetting, which is referenced by
1118 CIM_ConcreteComponent.

6.6.2.1.3.2 Pseudo Code

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

6.6.2.2 Show Multiple Instances – CIM_BootSourceSetting Reference

This command form is used when the `show` verb applies to multiple instances. This command form corresponds to a `show` command issued against instances of CIM_ConcreteComponent where only one reference is specified and the reference is to the instance of CIM_BootSourceSetting.

6.6.2.2.1 Command Form

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

6.6.2.2.2 CIM Requirements

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

6.6.2.2.3 Behavior Requirements

6.6.2.2.3.1 Preconditions

`$instance` represents the instance of CIM_BootSourceSetting, which is referenced by CIM_ConcreteComponent.

6.6.2.2.3.2 Pseudo Code

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

6.6.2.3 Show a Single Instance – CIM_BootSettingData Reference

This command form is used when the `show` verb applies to instances of CIM_ConcreteComponent where only one reference is specified and the reference is to an instance of a concrete subclass of CIM_BootSettingData.

The *Boot Control Profile* imposes a cardinality of 1 on CIM_ConcreteComponent.GroupComponent, which references an instance of CIM_BootConfigSetting. Therefore, for a given instance of a concrete subclass of CIM_BootSettingData, a single instance of CIM_BootConfigSetting is found.

6.6.2.3.1 Command Form

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

6.6.2.3.2 CIM Requirements

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

6.6.2.3.3 Behavior Requirements

6.6.2.3.3.1 Preconditions

`$instance` represents the instance a concrete subclass of CIM_BootSettingData.

6.6.2.3.3.2 Pseudo Code

```
&smOpReferences ( $instance.getObjectPath(), “CIM_ConcreteComponent”, “PartComponent”,
   NULL, NULL, $reference[] );
&smShowInstance ( $reference->[0]), NULL );
&smEnd;
```

6.6.2.4 Show Multiple Instance – CIM_SettingData Reference

This command form is used when the `show` verb applies to instances of CIM_ConcreteComponent where only one reference is specified and the reference is to an instance of a concrete subclass of CIM_SettingData.

6.6.2.4.1 Command Form

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

6.6.2.4.2 CIM Requirements

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

6.6.2.4.3 Behavior Requirements

6.6.2.4.3.1 Preconditions

`$instance` represents the instance of a concrete subclass of CIM_SettingData.

6.6.2.4.3.2 Pseudo Code

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

6.6.2.5 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_ConcreteComponent where both references are specified and therefore the desired instance is unambiguously identified.

6.6.2.5.1 Command Form

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

6.6.2.5.2 CIM Requirements

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

6.6.2.5.3 Behavior Requirements

6.6.2.5.3.1 Preconditions

`$instanceA` represents the instance of CIM_BootConfigSetting or the instance of CIM_BootSourceSetting which is referenced by CIM_ConcreteComponent.

`$instanceB` represents the instance of CIM_BootSettingData or the instance of CIM_SettingData which is referenced by CIM_ConcreteComponent.

6.6.2.5.3.2 Pseudo Code

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

6.7 CIM_ConcreteDependency

The `cd` and `help` 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_ConcreteDependency

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 6.7.2.
Start	Not supported	
Stop	Not supported	

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

6.7.1 Ordering of Results

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

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

6.7.2 Show

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

6.7.2.1 Show Multiple Instances – CIM_BootSourceSetting Reference

This command form is used when the `show` verb applies to multiple instances. This command form corresponds to a `show` command issued against instances of CIM_ConcreteDependency where only one reference is specified and the reference is to the instance of CIM_BootSourceSetting.

6.7.2.1.1 Command Form

```
show <CIM_ConcreteDependency multiple instances>
```

6.7.2.1.2 CIM Requirements

See CIM_ConcreteDependency in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.7.2.1.3 Behavior Requirements

6.7.2.1.3.1 Preconditions

`$instance` represents the instance of CIM_BootSourceSetting which is referenced by CIM_ConcreteDependency.

6.7.2.1.3.2 Pseudo Code

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

6.7.2.2 Show Multiple Instances – CIM_LogicalDevice Reference

This command form is used when the `show` verb applies to instances of CIM_ConcreteDependency where only one reference is specified and the reference is to the instance of a concrete subclass of CIM_LogicalDevice.

6.7.2.2.1 Command Form

```
show <CIM_ConcreteDependency multiple instances>
```

6.7.2.2.2 CIM Requirements

See CIM_ConcreteDependency in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.7.2.2.3 Behavior Requirements

6.7.2.2.3.1 Preconditions

`$instance` represents the instance of CIM_LogicalDevice which is referenced by CIM_ConcreteDependency.

6.7.2.2.3.2 Pseudo Code

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

6.7.2.3 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_ConcreteDependency where both references are specified and therefore the desired instance is unambiguously identified.

6.7.2.3.1 Command Form

```
show <CIM_ConcreteDependency single instance>
```

6.7.2.3.2 CIM Requirements

See CIM_ConcreteDependency in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.7.2.3.3 Behavior Requirements

6.7.2.3.3.1 Preconditions

`$instanceA` represents the instance of CIM_BootSourceSetting which is referenced by CIM_ConcreteDependency.

`$instanceB` represents the instance of CIM_LogicalDevice which is referenced by CIM_ConcreteDependency.

6.7.2.3.3.2 Pseudo Code

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

6.8 CIM_ElementCapabilities

The `cd`, `help`, `version`, and `exit` 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_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 6.8.2.
Start	Not supported	
Stop	Not supported	

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

6.8.1 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.

6.8.2 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.

6.8.2.1.1 Show a Single Instance – CIM_BootService Reference

This command form is used to apply the `show` verb to an instance of CIM_ElementCapabilities where only one reference is specified and the reference is to the instance of CIM_BootService.

6.8.2.1.2 Command Form

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

6.8.2.1.3 CIM Requirements

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

6.8.2.1.4 Behavior Requirements

6.8.2.1.4.1 Preconditions

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

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

`#propertylist[]`is an array of mandatory non-key property names.

6.8.2.1.4.2 Pseudo Code

```
if ( false != #all) { #propertylist[] = NULL; }
&smShowAssociationInstances ( “CIM_ElementCapabilities”, $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.8.2.2 Show Command Form for Multiple Instances – CIM_BootServiceCapabilities 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_BootServiceCapabilities.

6.8.2.2.1 Command Form

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

6.8.2.2.2 CIM Requirements

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

6.8.2.2.3 Behavior Requirements

6.8.2.2.3.1 Preconditions

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

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

`#propertylist[]`is an array of mandatory non-key property names.

6.8.2.2.3.2 Pseudo Code

```
if ( false != #all) { #propertylist[] = NULL; }
&smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath(),
   #propertylist[]);
&smEnd;
```

6.8.2.3 Show Command Form for a Single Instance Target – CIM_BootService and CIM_BootServiceCapabilities 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.

6.8.2.3.1 Command Form

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

6.8.2.3.2 CIM Requirements

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

6.8.2.3.3 Behavior Requirements

6.8.2.3.3.1 Preconditions

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

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

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

`#propertylist[]`is an array of mandatory non-key property names.

6.8.2.3.3.2 Pseudo Code

```
if ( false != #all) { #propertylist[] = NULL; }
&smShowAssociationInstance ( "CIM_ElementCapabilities", $instanceA.getObjectPath(),
   $instanceB.getObjectPath(), #propertylist[] );
&smEnd;
```

6.9 CIM_ElementSettingData

The `cd` and `help` 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_ElementSettingData

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	May	See 6.9.2.
show	Shall	See 6.9.3.
start	Not supported	
stop	Not supported	

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

6.9.1 Ordering of Results

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

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

6.9.2 Set

This section describes how to implement the `set` verb when applied to an instance of CIM_ElementSettingData. Implementations may support the use of the `set` verb for an instance of CIM_ElementSettingData which references an instance of CIM_BootConfigSetting.

6.9.2.1 Set of IsNext Property

This command form is for when the `set` verb is used to change the value of the `IsNext` property of an instance of CIM_ElementSettingData that associates an instance of CIM_ComputerSystem with an instance of CIM_BootConfigSetting. The valid input values are 1 (Is Next), 2 (Is Not Next), or 3 (Is Next for Single Use).

6.9.2.1.1 Command Form

```
set <CIM_ElementSettingData single instance> IsNext=<propertyvalue>
```

6.9.2.1.2 CIM Requirements

See CIM_ElementSettingData in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.9.2.1.3 Behavior Requirements

6.9.2.1.3.1 Preconditions

`$instance` represents the instance of CIM_ElementSettingData.

The property value specified in the command is one of the following valid values: 1 (Is Next), 2 (Is Not Next), or 3 (Is Next for Single Use).

`#intpropertyvalue` is the integer value of the <propertyvalue>.

6.9.2.1.3.2 Pseudo Code

```
if (#intpropertyvalue != 1 && #intpropertyvalue != 2 && #intpropertyvalue != 3) {
   $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 = 0x0000000E;
   $OperationError.Message = “The value specified for the {1} property is not valid.”;
   $OperationError.MessageArguments = { “IsNext” };
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
   &smEnd;
}
#propertyNames[] = “IsNext”
#propertyValues[] = #intpropertyvalue
&smSetInstance($instance, #propertyNames, #propertyValues);
&smEnd;
```

6.9.2.2 Set of IsCurrent Property

This command form is for when the `set` verb is used to change the value of the IsCurrent property of an instance of CIM_ElementSettingData that associates an instance of CIM_ComputerSystem with an instance of CIM_BootConfigSetting. The valid input value is 1 (Is Current).

This command form causes the Boot Configurable System to start the boot process, which applies the Next Boot Configuration or Next Single Use Boot Configuration. The command may be used when the boot process is started automatically as part of the system start or reset.

6.9.2.2.1 Command Form

```
set <CIM_ElementSettingData single instance> IsCurrent=<propertyvalue>
```

6.9.2.2.2 CIM Requirements

See CIM_ElementSettingData in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.9.2.2.3 Behavior Requirements

6.9.2.2.3.1 Preconditions

`$instance` represents the instance of CIM_ElementSettingData.

The property value specified in the command is the valid value, 1 (Is Current).

`#intpropertyvalue` is the integer value of the <propertyvalue>.

6.9.2.2.3.2 Pseudo Code

```
if (#intpropertyvalue != 1) {
   $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 = 0x0000000E;
   $OperationError.Message = “The value specified for the {1} property is not valid.”;
   $OperationError.MessageArguments = { “IsCurrent” };
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
   &smEnd;
}
// Try to find the CIM_BootService, take the first instance found
   $Services[] = smOpAssociators($instance.ManagedElement,
      “CIM_ServiceAffectsElement”, “CIM_BootService”, NULL, NULL);
   $Service-> = $Services[0].getObjectPath();
//Step 6, build parameter lists for method invocation
%InArguments[] = {
   newArgument(“BootConfigurableSystem”, $instance.ManagedElement)
   newArgument(“ApplyBootConfig”, $instance.SettingData)
};
%OutArguments[] = { newArgument( “Job”, instanceConcreteJob.getObjectPath() ) };
//step 7, invoke method
#returnStatus = smOpInvokeMethod ($Service->,
   “ApplyBootConfigSetting”,
   %InArguments[],
   %OutArguments[]);
//step 8, process return code to CLP Command Status
if (0 != #Error.code)  {
   //method invocation failed
   if ( (null != #Error.$error) && (null != #Error.$error[0]) )    {
      //if the method invocation contains an embedded error
      //use it for the Error for the overall job
```

```
      &smAddError($job, #Error.$error[0]);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else if ( 17 == #Error.code ) {
            //17 – CIM_ERR_METHOD_NOT_FOUND
            // The specified extrinsic method does not exist.
            $OperationError = smNewInstance("CIM_Error");
            // CIM_ERR_METHOD_NOT_FOUND
            $OperationError.CIMStatusCode = 17;
            //Software Error
            $OperationError.ErrorType = 10;
            //Unknown
            $OperationError.PerceivedSeverity = 0;
            $OperationError.OwningEntity = DMTF:SMCLP;
            $OperationError.MessageID = 0x00000001;
            $OperationError.Message = "Operation is not supported."
            &smAddError($job, $OperationError);
            &smMakeCommandStatus($job);
            &smEnd;
   }
   else {
      //operation failed, but no detailed error instance, need to make one up
      //make an Error instance and associate with job for Operation
         $OperationError = smNewInstance("CIM_Error");
         //CIM_ERR_FAILED
         $OperationError.CIMStatusCode = 1;
         //Software Error
         $OperationError.ErrorType = 4;
         //Unknown
         $OperationError.PerceivedSeverity = 0;
         $OperationError.OwningEntity = DMTF:SMCLP;
         $OperationError.MessageID = 0x00000009;
         $OperationError.Message = "An internal software error has occurred.";
         &smAddError($job, $OperationError);
         &smMakeCommandStatus($job);
         &smEnd;
   }
}//if CIM op failed
else if (0 == #returnStatus)  {
   //completed successfully
   &smCommandCompleted($job);
   &smEnd;
}
else if (4096 == #returnStatus)   {
   //job spawned, need to watch for it to finish
   //while the jobstate is "Running"
   while (4 == $instanceConcreteJob.JobState){<busy wait>}
   if (2 != $job.OperationalStatus) {
      %InArguments[] = { }
      %OutArguments[] = {newArgument("Job", $instanceConcreteJob.getObjectPath())}
```

```
      #Error = smOpInvokeMethod($job,
                "GetError"
                %InArguments,
                %OutArguments,
                #returncode);
      //Method invocation failed, internal processing error
      if ( (0 != #Error.code) || (0 != #returncode) )  {
      //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;
      }
      else  {
         //make command status
         $joberror = %OutArguments["Error"];
         &smMakeCommandExecutionFailed($job, {$joberror};
      }//end if have CIM_Error from GetError()
   }//embedded job not OK
}
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;
}
```

6.9.3 Show

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

6.9.3.1 Show a Single Instance – CIM_BootConfigSetting Reference

This command form is used when the `show` verb applies to instances of CIM_ElementSettingData where only one reference is specified and the reference is to an instance of CIM_BootConfigSetting.

The *Boot Control Profile* imposes a cardinality of 1 on CIM_ElementSettingData.ManagedElement, which references an instance of CIM_ComputerSystem. Therefore, for a given instance of CIM_BootConfigSetting, a single instance of CIM_ComputerSystem is found.

6.9.3.1.1 Command Form

```
show <CIM_ElementSettingData single instance>
```

6.9.3.1.2 CIM Requirements

See CIM_ElementSettingData in the "CIM Elements" section of the *Boot Control Profile* for the list of mandatory properties.

6.9.3.1.3 Behavior Requirements

6.9.3.1.3.1 Preconditions

`$instance` represents the instance of CIM_BootConfigSetting, which is referenced by CIM_ElementSettingData.

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

`#propertylist[]`is an array of mandatory non-key property names.

6.9.3.1.3.2 Pseudo Code

```
if (false != #all) { #propertylist[] = NULL; }
&smShowAssociationInstances ( "CIM_ElementSettingData", $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.9.3.2 Show Multiple Instance – CIM_ComputerSystem Reference

This command form is used when the `show` verb applies to instances of CIM_ElementSettingData where only one reference is specified and the reference is to the instance of CIM_ComputerSystem.

6.9.3.2.1 Command Form

```
show <CIM_ElementSettingData multiple instances>
```

6.9.3.2.2 CIM Requirements

See CIM_ElementSettingData in the "CIM Elements" section of the *Boot Control Profile* for the list of mandatory properties.

6.9.3.2.3 Behavior Requirements

6.9.3.2.3.1 Preconditions

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

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

`#propertylist[]`is an array of mandatory non-key property names.

6.9.3.2.3.2 Pseudo Code

```
if (false != #all) { #propertylist[] = NULL; }
&smShowAssociationInstances ( “CIM_ElementSettingData”, $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.9.3.3 Show a Single Instance – Both References

This command form is used when the `show` verb applies to a single instance. This command form corresponds to a `show` command issued against CIM_ElementSettingData where both references are specified and therefore the desired instance is unambiguously identified.

6.9.3.3.1 Command Form

```
show <CIM_ElementSettingData single instance>
```

6.9.3.3.2 CIM Requirements

See CIM_ElementSettingData in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.9.3.3.3 Behavior Requirements

6.9.3.3.3.1 Preconditions

`$instanceA` represents the referenced instance of CIM_ComputerSystem through CIM_ElementSettingData association.

`$instanceB` represents the other instance of CIM_BootConfigSetting which is referenced by CIM_ElementSettingData.

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

`#propertylist[]`is an array of mandatory non-key property names.

6.9.3.3.3.2 Pseudo Code

```
if (false != #all) { #propertylist[] = NULL; }
&smShowAssociationInstance ( “CIM_ElementSettingData”, $instanceA.getObjectPath(),
   $instanceB.getObjectPath(), #propertylist[] );
&smEnd;
```

6.10 CIM_HostedService

The `cd` and `help` 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.

1621 **Table 10 – 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 6.10.2.
Start	Not supported	
Stop	Not supported	

1622 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `exit`,
1623 `load`, `reset`, `set`, `start`, and `stop`.

1624 ### 6.10.1 Ordering of Results

1625 When results are returned for multiple instances of CIM_HostedService, implementations shall utilize the
1626 following algorithm to produce the natural (that is, default) ordering:

1627 - Results for CIM_HostedService are unordered; therefore, no algorithm is defined.

1628 ### 6.10.2 Show

1629 This section describes how to implement the `show` verb when applied to an instance of
1630 CIM_HostedService. Implementations shall support the use of the `show` verb with CIM_HostedService.

1631 #### 6.10.2.1 Show Multiple Instances – CIM_ComputerSystem Reference

1632 This command form applies the `show` verb to multiple instances. This command form corresponds to a
1633 `show` command issued against instances of CIM_HostedService where only one reference is specified
1634 and the reference is to an instance of CIM_ComputerSystem.

1635 ##### 6.10.2.1.1 Command Form

1636 **`show <CIM_HostedService` *`multiple instances`*`>`**

1637 ##### 6.10.2.1.2 CIM Requirements

1638 See CIM_HostedService in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory
1639 properties.

1640 ##### 6.10.2.1.3 Behavior Requirements

1641 ###### 6.10.2.1.3.1 Preconditions

1642 `$instance` represents the instance of CIM_ComputerSystem, which is referenced by
1643 CIM_HostedService.

6.10.2.1.3.2 Pseudo Code

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

6.10.2.2 Show a Single Instance – CIM_BootService Reference

This command form is used when the `show` verb applies to instances of CIM_HostedService where only one reference is specified and the reference is to an instance of CIM_BootService.

The *Boot Control Profile* imposes a cardinality of 1 on CIM_HostedService.Dependent, which references an instance of CIM_ComputerSystem. Therefore, for a given instance of CIM_BootService, a single instance of CIM_HostedService is found.

6.10.2.2.1 Command Form

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

6.10.2.2.2 CIM Requirements

See CIM_HostedService in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.10.2.2.3 Behavior Requirements

6.10.2.2.3.1 Preconditions

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

6.10.2.2.3.2 Pseudo Code

```
&smOpReferences ( $instance.getObjectPath(), “CIM_HostedService”, “Antecedent”, NULL,
   NULL, $references[] );
&smShowInstance ( $references->[0]), NULL );
&smEnd;
```

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

6.10.2.3.1 Command Form

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

6.10.2.3.2 CIM Requirements

See CIM_HostedService in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.10.2.3.3 Behavior Requirements

6.10.2.3.3.1 Preconditions

`$instanceA` represents the referenced instance of CIM_ComputerSystem through CIM_HostedService association.

`$instanceB` represents the other instance of CIM_BootService which is referenced by CIM_HostedService.

6.10.2.3.3.2 Pseudo Code

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

6.11 CIM_LogicalIdentity

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

Table 11 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 11 is for informational purposes only; in case of a conflict between Table 11 and requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 11.

Table 11 – Command Verb Requirements for CIM_LogicalIdentity

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 6.11.2
Start	Not supported	
Stop	Not supported	

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

6.11.1 Ordering of Results

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

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

6.11.2 Show

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

6.11.2.1 Show a Single Instance – CIM_BootSourceSetting Reference

This command form is used when the `show` verb applies to instances of CIM_LogicalIdentity where only one reference is specified and the reference is to an instance of CIM_BootSourceSetting.

The *Boot Control Profile* imposes a cardinality of zero or 1 on CIM_LogicalIdentity.SameElement, which references an instance of CIM_BootConfigSetting. Therefore, for a given instance of CIM_BootSourceSetting, a single instance of CIM_LogicalIdentity is found.

6.11.2.1.1 Command Form

```
show <CIM_LogicalIdentity single instance>
```

6.11.2.1.2 CIM Requirements

See CIM_LogicalIdentify in the "CIM Elements" section of the *Boot Control Profile* for the list of mandatory properties.

6.11.2.1.3 Behavior Requirements

6.11.2.1.3.1 Preconditions

`$instance` represents the instance of CIM_BootSourceSetting, which is referenced by CIM_LogicalIdentity.

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

`#propertylist[]`is an array of mandatory non-key property names.

6.11.2.1.3.2 Pseudo Code

```
if (false != #all) { #propertylist[] = NULL; }
&smShowAssociationInstances ( "CIM_LogicalIdentity", $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.11.2.2 Show a Single Instance – CIM_BootConfigSetting Reference

This command form is used when the `show` verb applies to instances of CIM_LogicalIdentity where only one reference is specified and the reference is to an instance of CIM_BootConfigSetting.

The *Boot Control Profile* imposes a cardinality of zero or 1 on CIM_LogicalIdentity.SystemElement, which references an instance of CIM_BootSourceSetting. Therefore, for a given instance of CIM_BootConfigSetting, a single instance of CIM_LogicalIdentity is found.

6.11.2.2.1 Command Form

```
show <CIM_LogicalIdentity single instance>
```

6.11.2.2.2 CIM Requirements

See CIM_LogicalIdentify in the "CIM Elements" section of the *Boot Control Profile* for the list of mandatory properties.

6.11.2.2.3 Behavior Requirements

6.11.2.2.3.1 Preconditions

`$instance` represents the instance of CIM_BootConfigSetting, which is referenced by CIM_LogicalIdentity.

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

`#propertylist[]`is an array of mandatory non-key property names.

6.11.2.2.3.2 Pseudo Code

```
if (false != #all) { #propertylist[] = NULL; }
&smShowAssociationInstances ( "CIM_LogicalIdentity", $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.11.2.3 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_LogicalIdentity where both references are specified and therefore the desired instance is unambiguously identified.

6.11.2.3.1 Command Form

```
show <CIM_LogicalIdentity single instance>
```

6.11.2.3.2 CIM Requirements

See CIM_LogicalIdentify in the "CIM Elements" section of the *Boot Control Profile* for the list of mandatory properties.

6.11.2.3.3 Behavior Requirements

6.11.2.3.3.1 Preconditions

`$instanceA` represents the referenced instance of CIM_BootSourceSetting through CIM_LogicalIdentity association.

`$instanceB` represents the other instance of CIM_BootConfigSetting which is referenced by CIM_LogicalIdentity.

6.11.2.3.3.2 Pseudo Code

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

6.12 CIM_OrderedComponent

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

Table 12 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 12 is for informational purposes only; in case of a conflict between Table 12 and requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 12.

1772 **Table 12 – Command Verb Requirements for CIM_OrderedComponent**

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 6.12.2.
start	Not supported	
stop	Not supported	

1773 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `exit`,
1774 `load`, `reset`, `start`, and `stop`.

1775 ### 6.12.1 Ordering of Results

1776 When results are returned for multiple instances of CIM_OrderedComponent, implementations shall
1777 utilize the following algorithm to produce the natural (that is, default) ordering:

1778 - Results for CIM_OrderedComponent are unordered; therefore, no algorithm is defined.

1779 ### 6.12.2 Show

1780 This section describes how to implement the `show` verb when applied to an instance of
1781 CIM_OrderedComponent. Implementations shall support the use of the `show` verb with
1782 CIM_OrderedComponent.

1783 #### 6.12.2.1 Show Multiple Instances – CIM_BootConfigSetting Reference

1784 This command form is used when the `show` verb applies to multiple instances. This command form
1785 corresponds to a `show` command issued against instances of CIM_OrderedComponent where only one
1786 reference is specified and the reference is to the instance of CIM_BootConfigSetting.

1787 ### 6.12.3 Command Form

1788 **`show <CIM_OrderedComponent`** ***`multiple instances`*****`>`**

1789 #### 6.12.3.1.1 CIM Requirements

1790 See CIM_OrderedComponent in the “CIM Elements” section of the *Boot Control Profile* for the list of
1791 mandatory properties.

1792 #### 6.12.3.1.2 Behavior Requirements

1793 #### 6.12.3.1.2.1 Preconditions

1794 `$instance` represents the instance of CIM_BootConfigSetting, which is referenced by
1795 CIM_OrderedComponent.

1796 `#all` is true, if the “-all” option was specified with the command; otherwise, `#all` is false.

1797 `#propertylist[]`is an array of mandatory non-key property names.

6.12.3.1.2.2 Pseudo Code

```
if (false != #all) { #propertylist[] = NULL; }
&smShowAssociationInstances ( “CIM_OrderedComponent”, $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.12.3.2 Show a Single Instance – CIM_BootSourceSetting Reference

This command form is used when the `show` verb applies to a single instance. This command form corresponds to a `show` command issued against instances of CIM_OrderedComponent where only one reference is specified and the reference is to the instance of CIM_BootSourceSetting. An instance of CIM_BootSourceSetting is referenced by exactly one instance of CIM_BootConfigSetting. Therefore, a single instance is returned.

6.12.3.2.1 Command Form

```
show <CIM_OrderedComponent single instance>
```

6.12.3.2.2 CIM Requirements

See CIM_OrderedComponent in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.12.3.2.3 Behavior Requirements

6.12.3.2.3.1 Preconditions

`$instance` represents the instance of CIM_BootSourceSetting, which is referenced by CIM_OrderedComponent.

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

`#propertylist[]`is an array of mandatory non-key property names.

6.12.3.2.3.2 Pseudo Code

```
if (false != #all) { #propertylist[] = NULL; }
&smShowAssociationInstances ( “CIM_OrderedComponent”, $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.12.3.3 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_OrderedComponent where both references are specified and therefore the desired instance is unambiguously identified.

6.12.3.3.1 Command Form

```
show <CIM_OrderedComponent single instance>
```

6.12.3.3.2 CIM Requirements

See CIM_OrderedComponent in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.12.3.3.3 Behavior Requirements

6.12.3.3.3.1 Preconditions

$instanceA represents the referenced instance of CIM_BootConfigSetting through CIM_OrderedComponent association.

$instanceB represents the other instance of CIM_BootSourceSetting which is referenced by CIM_OrderedComponent.

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

#propertylist[]is an array of mandatory non-key property names.

6.12.3.3.3.2 Pseudo Code

```
if (false != #all) { #propertylist[] = NULL; }
&smShowAssociationInstance ( "CIM_OrderedComponent", $instanceA.getObjectPath(),
   $instanceB.getObjectPath(), #propertylist[] );
&smEnd;
```

6.13 CIM_ServiceAffectsElement

The cd and help verbs shall be supported as described in DSP0216.

Table 13 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 13 is for informational purposes only; in case of a conflict between Table 13 and requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 13.

Table 13 – Command Verb Requirements for CIM_ServiceAffectsElement

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 6.13.2.
Start	Not supported	
Stop	Not supported	

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

6.13.1 Ordering of Results

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

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

6.13.2 Show

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

6.13.2.1 Show Multiple Instances – CIM_BootService Reference

This command form is used when the `show` verb applies to multiple instances. This command form corresponds to a `show` command issued against instances of CIM_ServiceAffectsElement where only one reference is specified and the reference is to an instance of CIM_BootService.

6.13.2.1.1 Command Form

```
show <CIM_ServiceAffectsElement multiple instances>
```

6.13.2.1.2 CIM Requirements

See CIM_ServiceAffectsElement in the "CIM Elements" section of the *Boot Control Profile* for the list of mandatory properties.

6.13.2.1.3 Behavior Requirements

6.13.2.1.3.1 Preconditions

`$instance` represents the instance of CIM_BootService, which is referenced by CIM_ServiceAffectsElement.

6.13.2.1.3.2 Pseudo Code

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

6.13.2.2 Show a Single Instance – CIM_ComputerSystem Reference

This command form is used when the `show` verb applies to multiple instances. This command form corresponds to a `show` command issued against instances of CIM_ServiceAffectsElement where only one reference is specified and the reference is to an instance of CIM_ComputerSystem. An instance of CIM_BootService is referenced by exactly one instance of CIM_ComputerSystem. Therefore, a single instance is returned.

6.13.2.2.1 Command Form

```
show <CIM_ServiceAffectsElement single instance>
```

6.13.2.2.2 CIM Requirements

See CIM_ServiceAffectsElement in the "CIM Elements" section of the *Boot Control Profile* for the list of mandatory properties.

6.13.2.2.3 Behavior Requirements

6.13.2.2.3.1 Preconditions

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

6.13.2.2.3.2 Pseudo Code

```
&smShowAssociationInstance ( “CIM_ServiceAffectsElement”, $instance.getObjectPath() );
&smEnd;
```

6.13.2.3 Show a Single Instance – Both References

This command form is used when the `show` verb applies to a single instance. This command form corresponds to a `show` verb issued against an instance of CIM_ServiceAffectsElement where both references are specified and therefore the desired instance is unambiguously identified.

6.13.2.3.1 Command Form

```
show <CIM_ServiceAffectsElement single instance>
```

6.13.2.3.2 CIM Requirements

See CIM_ServiceAffectsElement in the “CIM Elements” section of the *Boot Control Profile* for the list of mandatory properties.

6.13.2.3.3 Behavior Requirements

6.13.2.3.3.1 Preconditions

`$instanceA` represents the referenced instance of CIM_BootService through CIM_ServiceAffectsElement association.

`$instanceB` represents the other instance of CIM_ComputerSystem which is referenced by CIM_ServiceAffectsElement.

6.13.2.3.3.2 Pseudo Code

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

ANNEX A
(informative)

Change Log

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