Document Number: DSP0817

Date: 2009-07-14

Version: 1.0.0

IP Interface Profile SM CLP Command Mapping Specification

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

CONTENTS

Tables

77

Foreword

The *IP Interface Profile SM CLP Command Mapping Specification* (DSP0817) was prepared by the Server Management Working Group.

Conventions

The pseudo-code conventions utilized in this document are the Recipe Conventions as defined in SNIA SMI-S 1.1.0, section 7.6.

Acknowledgements

The authors wish to acknowledge the following participants from the DTMF Server Management Working Group:

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

Introduction

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

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

IP Interface 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 *IP Interface Profile*.

2 Normative References

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

2.1 Approved References

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

DMTF DSP1036, *IP Interface Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1036_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

175 **4.5**
176 **SM**
177 Server Management

178 **4.6**
179 **SMI-S**
180 Storage Management Initiative Specification

181 **4.7**
182 **SNIA**
183 Storage Networking Industry Association

184 **4.8**
185 **UFsT**
186 User Friendly selection Tag

187 5 Recipes

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

190 • smStartRSC()
191 • smStopRSC()
192 • smResetRSC()
193 • smShowInstance()
194 • smShowInstances()
195 • smSetInstance()
196 • smShowAssociationInstances()
197 • smShowAssociationInstance()
198 • smNewInstance()
199 • smAddError()
200 • smCommandStatus()

201 This mapping does not define any recipes for local reuse.

202 6 Mappings

203 The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in
204 the *IP Interface Profile*. Requirements specified here related to support for a CLP verb for a particular
205 class are solely within the context of this profile.

206 6.1 CIM_BindsToLANEndpoint

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

208 Table 1 lists each SM CLP verb, the required level of support for the verb in conjunction with instances of
209 the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the
210 verb and target. Table 1 is for informational purposes only; in case of a conflict between Table 1 and

requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 1.

Table 1 – Command Verb Requirements for CIM_BindsToLANEndpoint

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

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

6.1.1 Ordering of Results

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

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

6.1.2 Show

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

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

6.1.2.1 Show Multiple Instances

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

6.1.2.1.1 Command Form

```
show <CIM_BindsToLANEndpoint multiple objects>
```

6.1.3 CIM Requirements

See CIM_BindsToLANEndpoint in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.1.3.1.1 Behavior Requirements

6.1.3.1.1.1 Preconditions

`$instance` contains the instance of CIM_LANEndpoint which is referenced by CIM_BindsToLANEndpoint.

6.1.3.1.1.2 Pseudo Code

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

6.1.3.2 Show a Single Instance – CIM_IPProtocolEndpoint Reference

This command form is for the `show` verb applied to a single instance. This command form corresponds to a `show` command issued against CIM_BindsToLANEndpoint where the reference specified is to an instance of CIM_IPProtocolEndpoint. An instance of CIM_IPProtocolEndpoint is referenced by exactly one instance of CIM_BindsToLANEndpoint. Therefore, a single instance will be returned.

6.1.3.2.1 Command Form

```
show <CIM_BindsToLANEndpoint single object>
```

6.1.3.2.2 CIM Requirements

See CIM_BindsToLANEndpoint in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.1.3.2.3 Behavior Requirements

6.1.3.2.3.1 Preconditions

`$instance` contains the instance of CIM_IPProtocolEndpoint which is referenced by CIM_BindsToLANEndpoint.

6.1.3.2.3.2 Pseudo Code

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

6.1.3.3 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_BindsToLANEndpoint where both references are specified and therefore the desired instance is unambiguously identified.

6.1.3.3.1 Command Form

```
show <CIM_BindsToLANEndpoint single object>
```

6.1.3.3.2 CIM Requirements

See CIM_BindsToLANEndpoint in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.1.3.3.3 Behavior Requirements

6.1.3.3.3.1 Preconditions

`$instanceA` contains the instance of CIM_LANEndpoint which is referenced by CIM_BindsToLANEndpoint.

`$instanceB` contains the instance of CIM_IPProtocolEndpoint which is referenced by CIM_BindsToLANEndpoint.

6.1.3.3.3.2 Pseudo Code

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

6.2 CIM_ElementCapabilities

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

Table 2 lists each SM CLP verb, the required level of support for the verb in conjunction with instances of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and target. Table 2 is for informational purposes only; in case of a conflict between Table 2 and requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 2.

Table 2 – Command Verb Requirements for CIM_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 6.2.2.
start	Not supported	
stop	Not supported	

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

6.2.1 Ordering of Results

When results are returned for multiple instances of CIM_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.

6.2.2 Show

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

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

6.2.2.1 Show Multiple Instances – CIM_EnabledLogicalElementCapabilities Reference

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

6.2.2.1.1 Command Form

```
show <CIM_ElementCapabilities multiple objects>
```

6.2.2.1.2 CIM Requirements

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

6.2.2.1.3 Behavior Requirements

6.2.2.1.3.1 Preconditions

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

6.2.2.1.3.2 Pseudo Code

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

6.2.2.2 Show a Single Instance – CIM_IPProtocolEndpoint Reference

This command form is for the `show` verb applied to a single instance. This command form corresponds to a `show` command issued against CIM_ElementCapabilities where the reference specified is to an instance of CIM_EnabledLogicalElementCapabilities. An instance of CIM_EnabledLogicalElementCapabilities is referenced by exactly one instance of CIM_ElementCapabilities. Therefore, a single instance will be returned.

6.2.2.2.1 Command Form

```
show <CIM_ElementCapabilities single object>
```

6.2.2.2.2 CIM Requirements

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

6.2.2.2.3 Behavior Requirements

6.2.2.2.3.1 Preconditions

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

6.2.2.2.3.2 Pseudo Code

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

6.2.2.3 Show a Single Instance – Both References

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

6.2.2.3.1 Command Form

```
show <CIM_ElementCapabilities single object>
```

6.2.2.3.2 CIM Requirements

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

6.2.2.3.3 Behavior Requirements

6.2.2.3.3.1 Preconditions

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

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

6.2.2.3.3.2 Pseudo Code

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

6.3 CIM_ElementSettingData

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 instances of 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_ElementSettingData

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	May	See 6.3.2.

Command Verb	Requirement	Comments
show	Shall	See 6.3.3.
start	Not supported	
stop	Not supported	

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

6.3.1 Ordering of Results

When results are returned for multiple instances of CIM_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.

6.3.2 Set

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

6.3.2.1 Set 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_IPAssignmentSettingData with an instance of CIM_IPProtocolEndpoint. The only valid input value is 1 (Is Current). The IsCurrent property is not modified directly. The mapping interprets this value as a request to assign the referenced CIM_IPAssignmentSettingData to the referenced CIM_IPProtocolEndpoint instance using the CIM_IPConfigurationService associated with the CIM_IPProtocolEndpoint.

6.3.2.1.1 Command Form

```
set [-force] <CIM_ElementSettingData single instance> IsCurrent=<propvalue>
```

6.3.2.1.2 CIM Requirements

See CIM_ElementSettingData in the “CIM Elements” section of the *IP Interface Profile* for the CIM_IPConfigurationService.ApplySettingToIPProtocolEndpoint property.

6.3.2.1.3 Behavior Requirements

6.3.2.1.3.1 Preconditions

`$bForce` is true if the “-force” option was specified.

Make sure an instance of CIM_IPAssignmentSettingData (and not a subclass) is referenced. If a subclass (such as StaticIPAssignmentSettingData) is referenced instead, it is not a valid target.

6.3.2.1.3.2 Pseudo Code

```
$instance=<CIM_ElementSettingData single instance>
// $bForce is true if force option was specified
// Make sure the property value specified in the command is valid
// 1 (“Is Current”)
if (<propvalue> != 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;
}
// Make sure an instance of CIM_IPAssignmentSettingData (and not a sub-class)
// is referenced, if it references a sub-class instead (like
// StaticIPAssignmentSettingData), its not a valid target
if ( ! ($instance.SettingData ISA CIM_IPAssignmentSettingData) )   {
   $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 \“IsCurrent\” property can not be modified for
      this instance.”
   $OperationError.MessageArguments = { “IsCurrent” };
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
   &smEnd;
}
// try to find the CIM_IPConfigurationService
$Services[] = smOpAssociators(
   $instance.ManagedElement,
   “CIM_ServiceAffectsElement”,
   “CIM_IPConfigurationService”,
   NULL,
   NULL);
//no service associated, so applying configuration is not supported
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-> = $Services[0].getObjectPath();
// if current configuration, force option is required
// value of 1 is "Is Current"
if ($instance.IsCurrent == 1 && !#force) {
   $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 = 0x0000000F;
   $OperationError.Message = “The selected configuration is already active.
      Use the force option to re-apply it.”;
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
   &smEnd;
}
//invoke the method
//Step 6, build parameter lists for method invocation
%InArguments[] = {newArgument(“Endpoint”, $instance.ManagedElement),
newArgument (“Configuration”, $instance.SettingData),
%OutArguments[] = { newArgument(“Job”,
                    instanceConcreteJob.getObjectPath()) };
//step 7, invoke method
#returnStatus = smOpInvokeMethod ($Service->,
   “ApplySettingToIPProtocolEndpoint”,
   %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;
}
```

6.3.3 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.

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

6.3.3.1 Show Multiple Instances – CIM_IPProtocolEndpoint Reference with Other Reference to CIM_IPAssignmentSettingData

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

6.3.3.1.1 Command Form

```
show <CIM_ElementSettingData multiple objects>
```

6.3.3.1.2 CIM Requirements

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

6.3.3.1.3 Behavior Requirements

6.3.3.1.3.1 Preconditions

$instance contains the instance of CIM_IPProtocolEndpoint which is referenced by CIM_ElementSettingData.

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

6.3.3.1.3.2 Pseudo Code

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

6.3.3.2 Show Multiple Instances – CIM_IPAssignmentSettingData Reference

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

6.3.3.2.1 Command Form

```
show <CIM_ElementSettingData multiple objects>
```

6.3.3.2.2 CIM Requirements

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

6.3.3.2.3 Behavior Requirements

6.3.3.2.3.1 Preconditions

`$instance` contains the instance of CIM_IPAssignmentSettingData which is referenced by CIM_ElementSettingData.

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

6.3.3.2.3.2 Pseudo Code

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

6.3.3.3 Show Multiple Instances – CIM_IPProtocolEndpoint Reference with Other Reference to CIM_StaticIPAssignmentSettingData

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

6.3.3.3.1 Command Form

```
show <CIM_ElementSettingData multiple objects>
```

6.3.3.3.2 CIM Requirements

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

6.3.3.3.3 Behavior Requirements

6.3.3.3.3.1 Preconditions

`$instance` contains the instance of CIM_IPProtocolEndpoint which is referenced by CIM_ElementSettingData.

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

6.3.3.3.3.2 Pseudo Code

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

6.3.3.4 Show Multiple Instances – CIM_StaticIPAssignmentSettingData Reference

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

6.3.3.4.1 Command Form

```
show <CIM_ElementSettingData multiple objects>
```

6.3.3.4.2 CIM Requirements

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

6.3.3.4.3 Behavior Requirements

6.3.3.4.3.1 Preconditions

`$instance` contains the instance of CIM_StaticIPAssignmentSettingData which is referenced by CIM_ElementSettingData.

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

6.3.3.4.3.2 Pseudo Code

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

6.3.3.5 Show a Single Instance – CIM_IPProtocolEndpoint and CIM_IPAssignmentSettingData 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_ElementSettingData where both references are specified and therefore the desired instance is unambiguously identified.

6.3.3.5.1 Command Form

```
show <CIM_ElementSettingData single object>
```

6.3.3.5.2 CIM Requirements

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

6.3.3.5.3 Behavior Requirements

6.3.3.5.3.1 Preconditions

`$instanceA` contains the instance of CIM_IPProtocolEndpoint which is referenced by CIM_ElementSettingData.

$instanceB contains the instance of CIM_IPAssignmentSettingData referenced by CIM_ElementSettingData.

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

6.3.3.5.3.2 Pseudo Code

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

6.3.3.6 Show a Single Instance – CIM_IPProtocolEndpoint and CIM_StaticIPAssignmentSettingData 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_ElementSettingData where both references are specified and therefore the desired instance is unambiguously identified.

6.3.3.6.1 Command Form

```
show <CIM_ElementSettingData single object>
```

6.3.3.6.2 CIM Requirements

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

6.3.3.6.3 Behavior Requirements

6.3.3.6.3.1 Preconditions

$instanceA contains the instance of CIM_IPProtocolEndpoint which is referenced by CIM_ElementSettingData.

$instanceB contains the instance of CIM_StaticIPAssignmentSettingData referenced by CIM_ElementSettingData.

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

6.3.3.6.3.2 Pseudo Code

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

6.4 CIM_EnabledLogicalElementCapabilities

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 instances of 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_EnabledLogicalElementCapabilities

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.4.2.
start	Not supported	
stop	Not supported	

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

6.4.1 Ordering of Results

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

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

6.4.2 Show

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

The `show` verb is used to display information about the capabilities.

6.4.2.1 Show a Single Instance

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

6.4.2.1.1 Command Form

```
show <CIM_EnabledLogicalElementCapabilities single object>
```

6.4.2.1.2 CIM Requirements

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

6.4.2.1.3 Behavior Requirements

6.4.2.1.3.1 Preconditions

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

6.4.2.1.3.2 Pseudo Code

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

6.4.2.2 Show Multiple Instances

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

6.4.2.2.1 Command Form

```
show <CIM_EnabledLogicalElementCapabilities multiple objects>
```

6.4.2.2.2 CIM Requirements

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

6.4.2.2.3 Behavior Requirements

6.4.2.2.3.1 Preconditions

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

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

6.4.2.2.3.2 Pseudo Code

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

6.5 CIM_HostedAccessPoint

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

Table 5The following table lists each SM CLP verb, the required level of support for the verb in conjunction with instances of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and target. Table 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_HostedAccessPoint

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

6.5.1 Ordering of Results

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

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

6.5.2 Show

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

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

6.5.2.1 Show Multiple Instances – CIM_ComputerSystem Reference

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

6.5.2.1.1 Command Form

```
show <CIM_HostedAccessPoint multiple objects>
```

6.5.2.1.2 CIM Requirements

See CIM_HostedAccessPoint in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.5.2.1.3 Behavior Requirements

6.5.2.1.3.1 Preconditions

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

6.5.2.1.3.2 Pseudo Code

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

6.5.2.2 Show a Single Instance – CIM_IPProtocolEndpoint Reference

This command form is for the `show` verb applied to a single instance. This command form corresponds to a `show` command issued against CIM_HostedAccessPoint where the reference specified is to an instance of CIM_CIM_IPProtocolEndpoint. An instance of CIM_CIM_IPProtocolEndpoint is referenced by exactly one instance of CIM_HostedAccessPoint. Therefore, a single instance will be returned.

6.5.2.2.1 Command Form

```
show <CIM_HostedAccessPoint single object>
```

6.5.2.2.2 CIM Requirements

See CIM_HostedAccessPoint in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.5.2.2.3 Behavior Requirements

6.5.2.2.3.1 Preconditions

`$instance` contains the instance of CIM_CIM_IPProtocolEndpoint which is referenced by CIM_HostedAccessPoint.

6.5.2.2.3.2 Pseudo Code

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

6.5.2.3 Show a Single Instance – CIM_CIM_RemoteServiceAccessPoint Reference

This command form is for the `show` verb applied to a single instance. This command form corresponds to a `show` command issued against CIM_HostedAccessPoint where the reference specified is to an instance of CIM_CIM_RemoteServiceAccessPoint. An instance of CIM_CIM_RemoteServiceAccessPoint is referenced by exactly one instance of CIM_HostedAccessPoint. Therefore, a single instance will be returned.

6.5.2.3.1 Command Form

```
show <CIM_HostedAccessPoint single object>
```

6.5.2.3.2 CIM Requirements

See CIM_HostedAccessPoint in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.5.2.3.3 Behavior Requirements

6.5.2.3.3.1 Preconditions

`$instance` contains the instance of CIM_CIM_RemoteServiceAccessPoint which is referenced by CIM_HostedAccessPoint.

6.5.2.3.3.2 Pseudo Code

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

6.5.2.4 Show a Single Instance – Both References (IPProtocolEndpoint)

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

6.5.2.4.1 Command Form

```
show <CIM_HostedAccessPoint single object>
```

6.5.2.4.2 CIM Requirements

See CIM_HostedAccessPoint in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.5.2.4.3 Behavior Requirements

6.5.2.4.3.1 Preconditions

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

`$instanceB` contains the instance of CIM_CIM_IPProtocolEndpoint which is referenced by CIM_HostedAccessPoint.

6.5.2.4.3.2 Pseudo Code

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

6.5.2.5 Show a Single Instance – Both References (RemoteServiceAccessPoint)

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

6.5.2.5.1 Command Form

```
show <CIM_HostedAccessPoint single object>
```

6.5.2.5.2 CIM Requirements

See CIM_HostedAccessPoint in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.5.2.5.3 Behavior Requirements

6.5.2.5.3.1 Preconditions

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

`$instanceB` contains the instance of CIM_CIM_RemoteServiceAccessPoint which is referenced by CIM_HostedAccessPoint.

6.5.2.5.3.2 Pseudo Code

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

6.6 CIM_HostedService

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 instances of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and target. Table 6 is for informational purposes only; in case of a conflict between Table 6 and requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 6.

Table 6 – Command Verb Requirements for CIM_HostedService

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.6.2.
start	Not supported	
stop	Not supported	

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

6.6.1 Ordering of Results

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

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

6.6.2 Show

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

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

6.6.2.1 Show Multiple Instances – CIM_ComputerSystem Reference

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

6.6.2.1.1 Command Form

```
show <CIM_HostedService multiple objects>
```

6.6.2.1.2 CIM Requirements

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

6.6.2.1.3 Behavior Requirements

6.6.2.1.3.1 Preconditions

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

6.6.2.1.3.2 Pseudo Code

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

6.6.2.2 Show a Single Instance – CIM_IPConfigurationService Reference

This command form is for the `show` verb applied to a single instance. This command form corresponds to a `show` command issued against CIM_HostedService where the reference specified is to an instance of CIM_IPConfigurationService. An instance of CIM_IPConfigurationService is referenced by exactly one instance of CIM_HostedService. Therefore, a single instance will be returned.

6.6.2.2.1 Command Form

```
show <CIM_HostedService single object>
```

6.6.2.2.2 CIM Requirements

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

6.6.2.2.3 Behavior Requirements

6.6.2.2.3.1 Preconditions

`$instance` contains the instance of CIM_IPConfigurationService which is referenced by CIM_HostedService.

955 #### 6.6.2.2.3.2 Pseudo Code

956 ```
&smShowAssociationInstances (“CIM_HostedService”, $instance.getObjectPath());
957 &smEnd;
```

958 ### 6.6.2.3 Show a Single Instance – Both References

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

962 #### 6.6.2.3.1 Command Form

963 ```
show <CIM_HostedService single object>
```

964 #### 6.6.2.3.2 CIM Requirements

965 See CIM_HostedService in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory
966 properties.

967 #### 6.6.2.3.3 Behavior Requirements

968 #### 6.6.2.3.3.1 Preconditions

969 `$instanceA` contains the instance of CIM_ComputerSystem which is referenced by
970 CIM_HostedService.

971 `$instanceB` contains the instance of CIM_IPConfigurationService which is referenced by
972 CIM_HostedService.

973 #### 6.6.2.3.3.2 Pseudo Code

974 ```
&smShowAssociationInstance (“CIM_HostedService”, $instanceA.getObjectPath(),
975    $instanceB.getObjectPath());
976 &smEnd;
```

977 ## 6.7 CIM_IPAssignmentSettingData

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

979 Table 7 lists each SM CLP verb, the required level of support for the verb in conjunction with instances of
980 the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the
981 verb and target. Table 7 is for informational purposes only; in case of a conflict between Table 7 and
982 requirements detailed in the following sections, the text detailed in the following sections supersedes the
983 information in Table 7.

984 **Table 7 – Command Verb Requirements for CIM_IPAssignmentSettingData**

| Command Verb | Requirement | Comments |
|---|---|---|
| create | Not supported | |
| delete | Not supported | |
| dump | Not supported | |
| load | Not supported | |
| reset | Not supported | |
| set | May | See 6.7.2. |
```

Command Verb	Requirement	Comments
show	Shall	See 6.7.3.
start	Not supported	
stop	Not supported	

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

6.7.1 Ordering of Results

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

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

6.7.2 Set

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

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

6.7.2.1 General Usage of Set for a Single Property

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

The requirement for supporting modification of a property using this command form shall be equivalent to the requirement for supporting modification of the property using the ModifyInstance operation as defined in the *IP Interface Profile*.

6.7.2.1.1 Command Form

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

6.7.2.1.2 CIM Requirements

See CIM_IPAssignmentSettingData in the “CIM Elements” section of the *IP Interface Profile* for the list of modifiable properties.

6.7.2.1.3 Behavior Requirements

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

6.7.2.2 General Usage of Set for Multiple Properties

This command form corresponds to the general usage of the `set` verb to modify multiple properties of a target instance where there is not an explicit relationship between the properties. This is the most common case.

The requirement for supporting modification of a property using this command form shall be equivalent to the requirement for supporting modification of the property using the ModifyInstance operation as defined in the *IP Interface Profile*.

6.7.2.2.1 Command Form

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

6.7.2.2.2 CIM Requirements

See CIM_IPAssignmentSettingData in the "CIM Elements" section of the *IP Interface Profile* for the list of mandatory properties.

6.7.2.2.3 Behavior Requirements

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

6.7.3 Show

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

The `show` verb is used to display information about CIM_IPAssignmentSettingData.

6.7.3.1 Show a Single Instance

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

6.7.3.1.1 Command Form

```
show <CIM_IPAssignmentSettingData single object>
```

6.7.3.1.2 CIM Requirements

See CIM_IPAssignmentSettingData in the "CIM Elements" section of the *IP Interface Profile* for the list of mandatory properties.

6.7.3.1.3 Behavior Requirements

6.7.3.1.3.1 Preconditions

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

6.7.3.1.3.2 Pseudo Code

```
$instance=<CIM_IPAssignmentSettingData single object>
#propertylist[] = NULL;
if (false == #all)
   {
   #propertylist[] = { “ElementName”, “AddressOrigin” }
   }
&smShowInstance ( $instance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.7.3.2 Show Multiple Instances

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

6.7.3.2.1 Command Form

```
show <CIM_IPAssignmentSettingData multiple objects>
```

6.7.3.2.2 CIM Requirements

See CIM_IPAssignmentSettingData in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.7.3.2.3 Behavior Requirements

6.7.3.2.3.1 Preconditions

`$containerInstance` contains the instance of CIM_ConcreteCollection for which related CIM_IPAssignmentSettingData instances are displayed. SM ME Addressing requires that the CIM_IPAssignmentSettingData instance be associated with an instance of CIM_ConcreteCollection via an instance of the CIM_MemberOfCollection association.

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

6.7.3.2.3.2 Pseudo Code

```
#propertylist[] = NULL;
if (false == #all)
   {
   #propertylist[] = { “ElementName”, “AddressOrigin” }
   }
&smShowInstances ( “CIM_IPAssignmentSettingData”, “CIM_OrderedComponent”,
   $containerInstance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.8 CIM_IPConfigurationService

The `cd` and `help` 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 instances of 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.

1091 **Table 8 – Command Verb Requirements for CIM_IPConfigurationService**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	May	See 6.8.2.
show	Shall	See 6.8.3.
start	Not supported	
stop	Not supported	

1092 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,
1093 `reset`, `start`, and `stop`.

1094 ### 6.8.1 Ordering of Results

1095 When results are returned for multiple instances of CIM_IPConfigurationService, implementations shall
1096 utilize the following algorithm to produce the natural (that is, default) ordering:

1097 - Results for CIM_IPConfigurationService are unordered; therefore, no algorithm is defined.

1098 ### 6.8.2 Set

1099 This section describes how to implement the `set` verb when applied to an instance of
1100 CIM_IPConfigurationService. Implementations may support the use of the `set` verb with
1101 CIM_IPConfigurationService.

1102 No properties of the CIM_IPConfigurationService instance are writeable via the intrinsic ModifyInstance
1103 operation. Therefore, the only command form specified is for requesting a state change on the instance
1104 via assignment to the RequestedState property.

1105 #### 6.8.2.1 General Usage of Set for a Single Property

1106 This command form corresponds to the general usage of the `set` verb to modify a single property of a
1107 target instance. This is the most common case.

1108 The requirement for supporting modification of a property using this command form shall be equivalent to
1109 the requirement for supporting modification of the property using the ModifyInstance operation as defined
1110 in the *IP Interface Profile*.

1111 ##### 6.8.2.1.1 Command Form

1112 **`set <CIM_IPConfigurationService single instance> <propertyname>=<propertyvalue>`**

1113 ##### 6.8.2.1.2 CIM Requirements

1114 See CIM_IPConfigurationService in the “CIM Elements” section of the *IP Interface Profile* for the list of
1115 modifiable properties.

6.8.2.1.3 Behavior Requirements

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

6.8.2.2 General Usage of Set for Multiple Properties

This command form corresponds to the general usage of the `set` verb to modify multiple properties of a target instance where there is not an explicit relationship between the properties. This is the most common case.

The requirement for supporting modification of a property using this command form shall be equivalent to the requirement for supporting modification of the property using the ModifyInstance operation as defined in the *IP Interface Profile*.

6.8.2.2.1 Command Form

```
set <CIM_IPConfigurationService single instance> <propertyname1>=<propertyvalue1>
   <propertyname n>=<propertyvalue n>
```

6.8.2.2.2 CIM Requirements

See CIM_IPConfigurationService in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.8.2.2.3 Behavior Requirements

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

6.8.3 Show

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

The `show` verb is used to display information about the CIM_IPConfigurationService instance.

6.8.3.1 Show a Single Instance

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

6.8.3.1.1 Command Form

```
show <CIM_IPConfigurationService single object>
```

6.8.3.1.2 CIM Requirements

See CIM_IPConfigurationService in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.8.3.1.3 Behavior Requirements

```
$instance=<CIM_IPConfigurationService single object>
#propertylist[] = NULL;
if (false == #all)
   {
   #propertylist[] = { “ElementName” }
   }
&smShowInstance ( $instance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.8.3.2 Show Multiple Instances

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

6.8.3.2.1 Command Form

```
show <CIM_IPConfigurationService multiple objects>
```

6.8.3.2.2 CIM Requirements

See CIM_IPConfigurationService in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.8.3.2.3 Behavior Requirements

6.8.3.2.3.1 Preconditions

`$containerInstance` contains the instance of CIM_ComputerSystem for which scoped CIM_IPConfigurationService instances are displayed. The *IP Interface Profile* requires that the CIM_IPConfigurationService instance be associated with its scoping system via an instance of the CIM_HostedService association.

6.8.3.2.3.2 Pseudo Code

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

6.9 CIM_IPProtocolEndpoint

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 instances of 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_IPProtocolEndpoint

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	May	See 6.9.2.
set	May	See 6.9.3.
show	Shall	See 6.9.3.2.4.
start	May	See 6.9.4.
stop	May	See 6.9.5.

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

6.9.1 Ordering of Results

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

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

6.9.2 Reset

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

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

6.9.2.1 Reset a Single Instance

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

6.9.2.1.1 Command Form

```
reset <CIM_IPProtocolEndpoint single object>
```

6.9.2.1.2 CIM Requirements

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

6.9.2.1.3 Behavior Requirements

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

6.9.3 Set

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

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

6.9.3.1 General Usage of Set for a Single Property

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

The requirement for supporting modification of a property using this command form shall be equivalent to the requirement for supporting modification of the property using the ModifyInstance operation as defined in the *IP Interface Profile*.

6.9.3.1.1 Command Form

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

6.9.3.1.2 CIM Requirements

See CIM_IPProtocolEndpoint in the “CIM Elements” section of the *IP Interface Profile* for the list of modifiable properties.

6.9.3.1.3 Behavior Requirements

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

6.9.3.2 General Usage of Set for Multiple Properties

This command form corresponds to the general usage of the `set` verb to modify multiple properties of a target instance where there is not an explicit relationship between the properties. This is the most common case.

The requirement for supporting modification of a property using this command form shall be equivalent to the requirement for supporting modification of the property using the ModifyInstance operation as defined in the *IP Interface Profile*.

6.9.3.2.1 Command Form

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

6.9.3.2.2 CIM Requirements

See CIM_IPProtocolEndpoint in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.9.3.2.3 Behavior Requirements

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

6.9.3.2.4 Show

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

The `show` verb is used to display information about the IP interface.

6.9.3.3 Show a Single Instance

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

6.9.3.3.1 Command Form

```
show <CIM_IPProtocolEndpoint single object>
```

6.9.3.3.2 CIM Requirements

See CIM_IPProtocolEndpoint in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.9.3.3.3 Behavior Requirements

6.9.3.3.3.1 Preconditions

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

6.9.3.3.3.2 Pseudo Code

```
$instance=<CIM_IPProtocolEndpoint single object>
#propertylist[] = NULL;
if (false == #all)
   {
   #propertylist[] = { “IPv4Address”, “SubnetMask”, “AddressOrigin” };
   }
&smShowInstance ( $instance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.9.3.4 Show Multiple Instances

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

6.9.3.4.1 Command Form

```
show <CIM_IPProtocolEndpoint multiple objects>
```

6.9.3.4.2 CIM Requirements

See CIM_IPProtocolEndpoint in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.9.3.4.3 Behavior Requirements

6.9.3.4.3.1 Preconditions

`$containerInstance` contains the instance of CIM_ComputerSystem for which scoped CIM_IPProtocolEndpoint instances are displayed. The *IP Interface Profile* requires that the CIM_IPProtocolEndpoint instance be associated with its scoping system via an instance of the CIM_HostedAccessPoint association.

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

6.9.3.4.3.2 Pseudo Code

```
#propertylist[] = NULL;
if (false == #all)
   {
   #propertylist[] = { “IPv4Address”, “SubnetMask”, “AddressOrigin” };
   }
&smShowInstances ( “CIM_IPProtocolEndpoint”, “CIM_HostedAccessPoint”,
   $containerInstance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.9.4 Start

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

The `start` verb is used to enable an IP interface.

6.9.4.1 Start a Single Instance

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

6.9.4.1.1 Command Form

```
start <CIM_IPProtocolEndpoint single object>
```

6.9.4.1.2 CIM Requirements

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

6.9.4.1.3 Behavior Requirements

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

6.9.5 Stop

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

The `stop` verb is used to disable an IP interface.

6.9.5.1 Stop a Single Instance

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

6.9.5.1.1 Command Form

```
stop <CIM_IPProtocolEndpoint single object>
```

6.9.5.1.2 CIM Requirements

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

6.9.5.1.3 Behavior Requirements

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

6.10 CIM_OrderedComponent

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 instances of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and target. Table 10 is for informational purposes only; in case of a conflict between Table 10 and requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 10.

Table 10 – Command Verb Requirements for CIM_OrderedComponent

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

Command Verb	Requirement	Comments
reset	Not supported	
set	Not supported	
show	Shall	See 6.10.2.
start	Not supported	
stop	Not supported	

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

6.10.1 Ordering of Results

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

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

6.10.2 Show

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

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

6.10.2.1 Show Multiple Instances – CIM_IPAssignmentSettingData Reference

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

6.10.2.1.1 Command Form

```
show <CIM_OrderedComponent multiple objects>
```

6.10.2.1.2 CIM Requirements

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

6.10.2.1.3 Behavior Requirements

6.10.2.1.3.1 Preconditions

`$instance` contains the instance of CIM_IPAssignmentSettingData which is referenced by CIM_OrderedComponent.

There is only a single property and it is always returned.

6.10.2.1.3.2 Pseudo Code

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

6.10.2.2 Show a Single Instance – CIM_IPAssignmentSettingData Subclass Reference

This command form is for the `show` verb applied to a single instance. This command form corresponds to a `show` command issued against CIM_OrderedComponent where the reference specified is to an instance of a subclass of CIM_IPAssignmentSettingData.

6.10.2.2.1 Command Form

```
show <CIM_OrderedComponent single object>
```

6.10.2.2.2 CIM Requirements

See CIM_OrderedComponent in the "CIM Elements" section of the *IP Interface Profile* for the list of mandatory properties.

6.10.2.2.3 Behavior Requirements

6.10.2.2.3.1 Preconditions

`$instance` contains the instance of a subclass of CIM_IPAssignmentSettingData which is referenced by CIM_OrderedComponent.

There is only a single property and it is always returned.

6.10.2.2.3.2 Pseudo Code

```
&smShowAssociationInstances ( "CIM_OrderedComponent", $instance.getObjectPath(),
   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_OrderedComponent where both references are specified and therefore the desired instance is unambiguously identified.

6.10.2.3.1 Command Form

```
show <CIM_OrderedComponent single object>
```

6.10.2.3.2 CIM Requirements

See CIM_OrderedComponent in the "CIM Elements" section of the *IP Interface Profile* for the list of mandatory properties.

6.10.2.3.3 Behavior Requirements

6.10.2.3.3.1 Preconditions

`$instanceA` contains the instance of CIM_IPAssignmentSettingData that is referenced by CIM_OrderedComponent.

`$instanceB` contains the instance of a subclass of CIM_IPAssignmentSettingData which is referenced by CIM_OrderedComponent.

There is only a single property and it is always returned.

6.10.2.3.3.2 Pseudo Code

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

6.11 CIM_RemoteAccessAvailableToElement

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 instances of 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_RemoteAccessAvailableToElement

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

6.11.1 Ordering of Results

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

- Results for CIM_RemoteAccessAvailableToElement 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_RemoteAccessAvailableToElement. Implementations shall support the use of the `show` verb with CIM_RemoteAccessAvailableToElement.

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

6.11.2.1 Show Multiple Instances – CIM_RemoteServiceAccessPoint Reference

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

6.11.2.1.1 Command Form

```
show <CIM_RemoteAccessAvailableToElement multiple objects>
```

6.11.2.1.2 CIM Requirements

See CIM_RemoteAccessAvailableToElement in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.11.2.1.3 Behavior Requirements

6.11.2.1.3.1 Preconditions

`$instance` contains the instance of CIM_RemoteServiceAccessPoint which is referenced by CIM_RemoteAccessAvailableToElement.

There is only a single property and it is always returned.

6.11.2.1.3.2 Pseudo Code

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

6.11.2.2 Show a Single Instance – CIM_IPProtocolEndpoint Reference

This command form is for the `show` verb applied to a single instance. This command form corresponds to a `show` command issued against CIM_RemoteAccessAvailableToElement where the reference specified is to an instance of CIM_IPProtocolEndpoint. An instance of CIM_IPProtocolEndpoint is referenced by exactly one instance of CIM_RemoteAccessAvailableToElement. Therefore, a single instance will be returned.

6.11.2.2.1 Command Form

```
show <CIM_RemoteAccessAvailableToElement single object>
```

6.11.2.2.2 CIM Requirements

See CIM_RemoteAccessAvailableToElement in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.11.2.2.3 Behavior Requirements

6.11.2.2.3.1 Preconditions

`$instance` contains the instance of CIM_IPProtocolEndpoint which is referenced by CIM_RemoteAccessAvailableToElement.

There is only a single property and it is always returned.

6.11.2.2.3.2 Pseudo Code

```
&smShowAssociationInstances ( “CIM_RemoteAccessAvailableToElement”,
   $instance.getObjectPath(), NULL );
&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_RemoteAccessAvailableToElement where both references are specified and therefore the desired instance is unambiguously identified.

6.11.2.3.1 Command Form

```
show <CIM_RemoteAccessAvailableToElement single object>
```

6.11.2.3.2 CIM Requirements

See CIM_RemoteAccessAvailableToElement in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.11.2.3.3 Behavior Requirements

6.11.2.3.3.1 Preconditions

`$instanceA` contains the instance of CIM_RemoteServiceAccessPoint which is referenced by CIM_RemoteAccessAvailableToElement.

`$instanceB` contains the instance of CIM_IPProtocolEndpoint which is referenced by CIM_RemoteAccessAvailableToElement.

There is only a single property and it is always returned.

6.11.2.3.3.2 Pseudo Code

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

6.12 CIM_RemoteServiceAccessPoint

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 instances of 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.

Table 12 – Command Verb Requirements for CIM_RemoteServiceAccessPoint

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

Command Verb	Requirement	Comments
reset	Not supported	
set	May	See 6.12.2.
show	Shall	See 6.12.3.
start	Not supported	
stop	Not supported	

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

6.12.1 Ordering of Results

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

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

6.12.2 Set

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

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

6.12.2.1 General Usage of Set for a Single Property

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

The requirement for supporting modification of a property using this command form shall be equivalent to the requirement for supporting modification of the property using the ModifyInstance operation as defined in the *IP Interface Profile*.

6.12.2.1.1 Command Form

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

6.12.2.1.2 CIM Requirements

See CIM_RemoteServiceAccessPoint in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.12.2.1.3 Behavior Requirements

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

6.12.2.2 General Usage of Set for Multiple Properties

This command form corresponds to the general usage of the `set` verb to modify multiple properties of a target instance where there is not an explicit relationship between the properties. This is the most common case.

The requirement for supporting modification of a property using this command form shall be equivalent to the requirement for supporting modification of the property using the ModifyInstance operation as defined in the *IP Interface Profile*.

6.12.2.2.1 Command Form

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

6.12.2.2.2 CIM Requirements

See CIM_RemoteServiceAccessPoint in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.12.2.2.3 Behavior Requirements

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

6.12.3 Show

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

The `show` verb is used to display information about the gateway.

6.12.3.1 Show a Single Instance

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

6.12.3.1.1 Command Form

```
show <CIM_RemoteServiceAccessPoint single object>
```

6.12.3.1.2 CIM Requirements

See CIM_RemoteServiceAccessPoint in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.12.3.1.3 Behavior Requirements

6.12.3.1.3.1 Preconditions

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

6.12.3.1.3.2 Pseudo Code

```
$instance=<CIM_RemoteServiceAccessPoint single object>
#propertylist[] = NULL;
if (false == #all)
   {
   #propertylist[] = { “AccessContext”, “AccessInfo”, “InfoFormat”, “ElementName” };
   }
&smShowInstance ( $instance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.12.3.2 Show Multiple Instances

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

6.12.3.2.1 Command Form

```
show <CIM_RemoteServiceAccessPoint multiple objects>
```

6.12.3.2.2 CIM Requirements

See CIM_RemoteServiceAccessPoint in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.12.3.2.3 Behavior Requirements

6.12.3.2.3.1 Preconditions

`$containerInstance` contains the instance of CIM_ComputerSystem for which scoped CIM_RemoteServiceAccessPoint instances are displayed. The *IP Interface Profile* requires that the CIM_RemoteServiceAccessPoint instance be associated with its scoping system via an instance of the CIM_HostedAccessPoint association.

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

6.12.3.2.3.2 Pseudo Code

```
#propertylist[] = NULL;
if (false == #all)
   {
   #propertylist[] = { “AccessContext”, “AccessInfo”, “InfoFormat”, “ElementName” };
   }
&smShowInstances ( “CIM_RemoteServiceAccessPoint”, “CIM_HostedAccessPoint”,
   $containerInstance.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 instances of 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.

1625 **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	

1626 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,
1627 `reset`, `set`, `start`, and `stop`.

1628 ### 6.13.1 Ordering of Results

1629 When results are returned for multiple instances of CIM_ServiceAffectsElement, implementations shall
1630 utilize the following algorithm to produce the natural (that is, default) ordering:

1631 - Results for CIM_ServiceAffectsElement are unordered; therefore, no algorithm is defined.

1632 ### 6.13.2 Show

1633 This section describes how to implement the `show` verb when applied to an instance of
1634 CIM_ServiceAffectsElement. Implementations shall support the use of the `show` verb with
1635 CIM_ServiceAffectsElement.

1636 The `show` command is used to display information about the CIM_ServiceAffectsElement instance or
1637 instances.

1638 #### 6.13.2.1 Show Multiple Instances – CIM_IPConfigurationService Reference

1639 This command form is for the `show` verb applied to multiple instances. This command form corresponds
1640 to a `show` command issued against CIM_ServiceAffectsElement where only one reference is specified
1641 and the reference is to an instance of CIM_IPConfigurationService.

1642 ##### 6.13.2.1.1 Command Form

1643 ```
show <CIM_ServiceAffectsElement multiple objects>
```

1644 ##### 6.13.2.1.2 CIM Requirements

1645 See CIM_ServiceAffectsElement in the “CIM Elements” section of the *IP Interface Profile* for the list of
1646 mandatory properties.

1647 ##### 6.13.2.1.3 Behavior Requirements

1648 ###### 6.13.2.1.3.1 Preconditions

1649 `$instance` contains the instance of CIM_IPConfigurationService which is referenced by
1650 CIM_ServiceAffectsElement.
```

There is only a single property and it is always returned.

6.13.2.1.3.2 Pseudo Code

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

6.13.2.2 Show a Single Instance – CIM_IPProtocolEndpoint Reference

This command form is for the `show` verb applied to a single instance. This command form corresponds to a `show` command issued against CIM_ServiceAffectsElement where the reference specified is to an instance of CIM_IPProtocolEndpoint. An instance of CIM_IPProtocolEndpoint is referenced by exactly one instance of CIM_ServiceAffectsElement. Therefore, a single instance will be returned.

6.13.2.2.1 Command Form

```
show <CIM_ServiceAffectsElement single object>
```

6.13.2.2.2 CIM Requirements

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

6.13.2.2.3 Behavior Requirements

6.13.2.2.3.1 Preconditions

`$instance` contains the instance of CIM_IPProtocolEndpoint which is referenced by CIM_ServiceAffectsElement.

There is only a single property and it is always returned.

6.13.2.2.3.2 Pseudo Code

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

6.13.2.3 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_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 object>
```

6.13.2.3.2 CIM Requirements

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

6.13.2.3.3 Behavior Requirements

6.13.2.3.3.1 Preconditions

`$instanceA` contains the instance of CIM_IPConfigurationService which is referenced by CIM_ServiceAffectsElement.

`$instanceB` contains the instance of CIM_IPProtocolEndpoint which is referenced by CIM_ServiceAffectsElement.

There is only a single property and it is always returned.

6.13.2.3.3.2 Pseudo Code

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

6.14 CIM_StaticIPAssignmentSettingData

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

Table 14 lists each SM CLP verb, the required level of support for the verb in conjunction with instances of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and target. Table 14 is for informational purposes only; in case of a conflict between Table 14 and requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 14.

Table 14 – Command Verb Requirements for CIM_StaticIPAssignmentSettingData

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	May	See 6.14.2.
show	Shall	See 6.14.3.
start	Not supported	
stop	Not supported	

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

6.14.1 Ordering of Results

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

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

6.14.2 Set

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

The `set` verb is used to modify descriptive properties of an instance of CIM_StaticIPAssignmentSettingData.

6.14.2.1 General Usage of Set for a Single Property

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

The requirement for supporting modification of a property using this command form shall be equivalent to the requirement for supporting modification of the property using the ModifyInstance operation as defined in the *IP Interface Profile*.

6.14.2.1.1 Command Form

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

6.14.2.1.2 CIM Requirements

See CIM_StaticIPAssignmentSettingData in the “CIM Elements” section of the *IP Interface Profile* for the list of modifiable properties.

6.14.2.1.3 Behavior Requirements

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

6.14.2.2 General Usage of Set for Multiple Properties

This command form corresponds to the general usage of the `set` verb to modify multiple properties of a target instance where there is not an explicit relationship between the properties. This is the most common case.

The requirement for supporting modification of a property using this command form shall be equivalent to the requirement for supporting modification of the property using the ModifyInstance operation as defined in the *IP Interface Profile*.

6.14.2.2.1 Command Form

```
set <CIM_StaticIPAssignmentSettingData single instance>
   <propertyname1>=<propertyvalue1> <propertyname n>=<propertyvalue n>
```

6.14.2.2.2 CIM Requirements

See CIM_StaticIPAssignmentSettingData in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.14.2.2.3 Behavior Requirements

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

6.14.3 Show

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

The `show` verb is used to display information about the CIM_StaticIPAssignmentSettingData instance.

6.14.3.1 Show a Single Instance

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

6.14.3.1.1 Command Form

```
show <CIM_StaticIPAssignmentSettingData single object>
```

6.14.3.1.2 CIM Requirements

See CIM_StaticIPAssignmentSettingData in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.14.3.1.3 Behavior Requirements

6.14.3.1.3.1 Preconditions

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

6.14.3.1.3.2 Pseudo Code

```
$instance=<CIM_StaticIPAssignmentSettingData single object>
#propertylist[] = NULL;
if (false == #all)
   {
   #propertylist[] = {“ElementName”, “IPv4Address”, “SubnetMask”,
      “GatewayIPv4Address”, “AddressOrigin” }
   }
&smShowInstance ( $instance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.14.3.2 Show Multiple Instances

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

6.14.3.2.1 Command Form

```
show <CIM_StaticIPAssignmentSettingData multiple objects>
```

6.14.3.2.2 CIM Requirements

See CIM_StaticIPAssignmentSettingData in the “CIM Elements” section of the *IP Interface Profile* for the list of mandatory properties.

6.14.3.2.3 Behavior Requirements

6.14.3.2.3.1 Preconditions

`$containerInstance` contains the instance of CIM_IPAssignmentSettingData for which scoped CIM_StaticIPAssignmentSettingData instances are displayed. The *IP Interface Profile* requires that the CIM_StaticIPAssignmentSettingData instance be associated with an instance of CIM_IPAssignmentSettingData via an instance of the CIM_OrderedComponent association.

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

6.14.3.2.3.2 Pseudo Code

```
#propertylist[] = NULL;
if (false == #all)
   {
   #propertylist[] = {“ElementName”, “IPv4Address”, “SubnetMask”,
      “GatewayIPv4Address”, “AddressOrigin” }
   }
&smShowInstances ( “CIM_StaticIPAssignmentSettingData”, “CIM_OrderedComponent”,
   $containerInstance.getObjectPath(), #propertylist[] );
&smEnd;
```

ANNEX A
(informative)

Change Log

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