1

2 **Document Number: DSP0830**

3 **Date: 2009-07-14**

4 **Version: 1.0.0**

5 # Role Based Authorization Profile SM CLP
6 # Command Mapping Specification

7 **Document Type: Specification**

8 **Document Status: DMTF Standard**

9 **Document Language: E**

10

13 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems
14 management and interoperability. Members and non-members may reproduce DMTF specifications and
15 documents, provided that correct attribution is given. As DMTF specifications may be revised from time to
16 time, the particular version and release date should always be noted.

17 Implementation of certain elements of this standard or proposed standard may be subject to third party
18 patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations
19 to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose,
20 or identify any or all such third party patent right, owners or claimants, nor for any incomplete or
21 inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to
22 any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize,
23 disclose, or identify any such third party patent rights, or for such party's reliance on the standard or
24 incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any
25 party implementing such standard, whether such implementation is foreseeable or not, nor to any patent
26 owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is
27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party
28 implementing the standard from any and all claims of infringement by a patent owner for such
29 implementations.

30 For information about patents held by third-parties which have notified the DMTF that, in their opinion,
31 such patent may relate to or impact implementations of DMTF standards, visit
32 http://www.dmtf.org/about/policies/disclosures.php.

CONTENTS

Tables

75

76   Foreword

77   The *Role Based Authorization Profile SM CLP Command Mapping Specification* (DSP0830) was
78   prepared by the Server Management Working Group.

Conventions

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

Acknowledgements

83   The authors wish to acknowledge the following participants from the DMTF Server Management Working
84   Group:

85   •   Khachatur Papanyan – Dell

86   •   Aaron Merkin – IBM

87   •   Jon Hass – Dell

88   •   Jeff Hilland – HP

89   •   Christina Shaw – HP

90   •   Aaron Merkin – IBM

91   •   Perry Vincent – Intel

92   •   John Leung – Intel

93

94   Introduction

95   This document defines the SM CLP mapping for CIM elements described in the *Role Based Authorization*
96   *Profile*. The information in this specification, combined with the *SM CLP-to-CIM Common Mapping*
97   *Specification 1.0* (DSP0216), is intended to be sufficient to implement SM CLP commands relevant to the
98   classes, properties, and methods described in the *Role Based Authorization Profile* using CIM operations.

99   The target audience for this specification is implementers of the SM CLP support for the *Role Based*
100   *Authorization Profile*.

Role Based Authorization 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 *Role Based Authorization 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 DSP1039, *Role Based Authorization Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1039_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

132 **3.4**
133 **mandatory**
134 indicates requirements to be followed strictly in order to conform to the document and from which no
135 deviation is permitted

136 **3.5**
137 **may**
138 indicates a course of action permissible within the limits of the document

139 **3.6**
140 **need not**
141 indicates a course of action permissible within the limits of the document

142 **3.7**
143 **optional**
144 indicates a course of action permissible within the limits of the document

145 **3.8**
146 **shall**
147 indicates requirements to be followed strictly in order to conform to the document and from which no
148 deviation is permitted

149 **3.9**
150 **shall not**
151 indicates requirements to be followed strictly in order to conform to the document and from which no
152 deviation is permitted

153 **3.10**
154 **should**
155 indicates that among several possibilities, one is recommended as particularly suitable, without
156 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required

157 **3.11**
158 **should not**
159 indicates that a certain possibility or course of action is deprecated but not prohibited

160 **4 Symbols and Abbreviated Terms**

161 The following symbols and abbreviations are used in this document.

162 **4.1**
163 **CIM**
164 Common Information Model

165 **4.2**
166 **CLP**
167 Command Line Protocol

168 **4.3**
169 **DMTF**
170 Distributed Management Task Force

171 **4.4**

172 **IETF**

173 Internet Engineering Task Force

174 **4.5**

175 **SM**

176 Server Management

177 **4.6**

178 **SMI-S**

179 Storage Management Initiative Specification

180 **4.7**

181 **SNIA**

182 Storage Networking Industry Association

183 **4.8**

184 **UFsT**

185 User Friendly selection Tag

186 # 5   Recipes

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

189 • smShowInstance( )

190 • smShowInstances( )

191 • smSetInstance( )

192 • smShowAssociationInstances( )

193 • smShowAssociationInstance( )

194 For convenience, Table 1 lists each recipe defined in this mapping which is used for more than one verb
195 or class mapping.

196   **Table 1 – Local Recipes**

Recipe Name	Description	Definition
IShowRole	Show an instance of CIM_Role.	See 5.1.
IModifyRole	Modify an instance of CIM_Role.	See 5.2.

197 The following sections detail Local Recipes defined for use in this mapping.

198 ## 5.1   IShowRole

199 ### 5.1.1   Description

200 Reusable recipe for displaying an instance of CIM_Role. A recipe is defined for re-use by the `show` and
201 `create` verbs applied to CIM_Role. The behavior of this function is as follows:

202   1)   Retrieve the CIM_Role instance.

203   2)   Find the associated instance of CIM_RoleBasedAuthorizationService.

204 3) Invoke ShowRoles().

205 4) Match target CIM_Role instance to embedded instance returned.

206 5) Find embedded instance of CIM_Privilege at corresponding array position.

207 6) Add the relevant CIM_Privilege properties to the CIM_Role instance as propagated properties.

208 7) Display the CIM_Role instance.

5.1.2 Pseudo Code

```
210  // $Role-> contains the object path of the CIM_Role to display
211  // #all indicates whether the all option was specified
212  sub lShowRole($Role->, #all)  {
213      #propertylist[] = NULL;
214      //if we're not displaying all of the properties, provide a list
215      if (false == #all)  {
216          #propertylist[] = { //all mandatory non-key properties };
217      }
218      //get the instance
219      #Error = &smGetInstance ( $Role->, $Role );
220      if (0 != #Error.code)      {
221               &smProcessOpError (#Error);
222               //includes &smEnd;
223      }
224  #Error = &smOpAssociators(
225          $Role->,
226          "CIM_ServiceAffectsElement",
227          "CIM_RoleBasedAuthorizationService",
228          NULL,
229          NULL,
230          NULL,
231          $SvcInstancePaths[])
232
233  if (0 != #Error.code)
234  {
235          &smProcessOpError (#Error);
236          //includes &smEnd;
237  }
238  // invoke the method
239  //build the parameter lists and invoke the method
240  %InArguments[] = { }
241  %OutArguments[] = { newArgument("Roles", #RoleStrings[]),
242                      newArgument("Privileges", #PrivilegeStrings[]->) };
243  //invoke method
244  #returnStatus = smOpInvokeMethod ($SvcInstancePaths[0].GetObjectPath(),
245      "ShowRoles",
246      %InArguments[],
247      %OutArguments[]);
248  //3 process return code to CLP Command Status
249  if (0 != #Error.code)  {
250      //method invocation failed
251      if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )    {
252          // if the method invocation contains an embedded error
253          // use it for the Error for the overall job
```

```
254        &smAddError($job, #Error.$error[0]);
255        &smMakeCommandStatus($job);
256        &smEnd;
257    }
258    else if (#Error.code == 17)    {
259    //trap for CIM_METHOD_NOT_FOUND
260    //and make nice Unsupported msg.
261        //unsupported
262        $OperationError = smNewInstance("CIM_Error");
263        //CIM_ERR_NOT_SUPPORTED
264        $OperationError.CIMStatusCode = 7;
265        //Other
266        $OperationError.ErrorType = 1;
267        //Low
268        $OperationError.PerceivedSeverity = 2;
269        $OperationError.OwningEntity = DMTF:SMCLP;
270        $OperationError.MessageID = 0x00000001;
271        $OperationError.Message = "Operation is not supported.";
272        &smAddError($job, $OperationError);
273        &smMakeCommandStatus($job);
274        &smEnd;
275    }
276    else if (#Error.code != 0) {
277        //operation failed, but no detailed error instance, need to make one up
278        //make an Error instance and associate with job for Operation
279            $OperationError = smNewInstance("CIM_Error");
280            //CIM_ERR_FAILED
281            $OperationError.CIMStatusCode = 1;
282            //Software Error
283            $OperationError.ErrorType = 4;
284            //Unknown
285            $OperationError.PerceivedSeverity = 0;
286            $OperationError.OwningEntity = DMTF:SMCLP;
287            $OperationError.MessageID = 0x00000009;
288            $OperationError.Message = "An internal software error has occurred.";
289            &smAddError($job, $OperationError);
290            &smMakeCommandStatus($job);
291            &smEnd;
292    }
293 }//if CIM op failed
294    //$Roles[] contains instances of CIM_Role generated from the embedded instances
295        contained in #RoleStrings[]
296    //$Privileges contains instances of CIM_Privilege generated from the embedded
297        instances contained in #PrivilegeStrings[]
298    //placeholder for privilege instance if one is found.
299    $Privilege = NULL;
300    for (#i-0; i<$Roles[].length; i++ {
301        if ($Roles[#i].CommonName = $Role.CommonName) {
302            $Privilege = $Privileges[#i];
303            break;
304        }
305    }
306    #referencedproperties[] = NULL;
```

```
307     if (NULL != $Privilege){
308         $Role.Activities = $Privilege.Activities;
309         $Role.ActivityQualifiers = $Privilege.ActivityQualifiers;
310         $Role.QualifierFormats = $Privilege.QualifierFormats;
311         #referencedProperties[] =  { "Activities", "ActivityQualifiers",
312             "QualifierFormats" };
313     }
314     &smShowInstancePseudoProperties ( $Role.GetObjectPath(), #propertyList[],
315         #referencedProperties[] );
316     &smEnd;
317 } //lShowRole()
```

5.2   lModifyRole

5.2.1   Description

Reusable recipe for modifying an instance of CIM_Role. This recipe is defined for re-use by the set
verbs applied to CIM_Role. The behavior of this function is as follows:

1) Find the associated instance of CIM_RoleBasedAuthorizationService.

2) Create embedded instance of CIM_Privilege from propagated properties.

3) Invoke ModifyRole() – if any propagated properties.

4) Invoke smModifyInstance() if any intrinsic properties.

5) Call lShowRole().

5.2.2   Pseudo Code

```
328 // $Role-> contains the object path of the CIM_Role to modify
329 // #propertyNames[] contains array of property names specified on the command line
330 //#propertyValues[] contains array of property values assigned for each property name
331 // #all indicates whether the all option was specified
332 sub lModifyRole($Role->, #propertyNames[], #propertyValues[], #all){
333     //get the instance
334     #Error = &smGetInstance ( $Role->, $Role );
335     if (0 != #Error.code)       {
336             &smProcessOpError (#Error);
337             //includes &smEnd;
338     }
339 #Error = &smOpAssociators(
340         $Role->,
341         "CIM_ServiceAffectsElement",
342         "CIM_RoleBasedAuthorizationService",
343         NULL,
344         NULL,
345         NULL,
346         $SvcInstancePaths[])
347 if (0 != #Error.code)
348 {
349     &smProcessOpError (#Error);
350     //includes &smEnd;
351 }
```

```
352    // invoke the method
353    //build the parameter lists and invoke the method
354    $PrivilegeTemplate = smNewInstance ("CIM_Privilege");
355    $PrivilegeTemplate.PrivilegesGranted = TRUE;
356    #atLeastOnePrivilege = FALSE;
357    #atLeastOneRole = FALSE;
358    for (#i < #propertynames[].length)
359    {
360       if (#propertynames[#i] == "Activities")  {
361          $PrivilegeTemplate.Activities = #propertyvalues[#i];
362          #atLeastOnePrivilege=TRUE;
363       }
364       else if (#propertynames[#i] == "ActivityQualifiers") {
365       $PrivilegeTemplate.ActivityQualifiers = #propertyvalues[#i];
366          #atLeastOnePrivilege=TRUE;
367       }
368       else if (#propertynames[#i] == "QualifierFormats")   {
369          #atLeastOnePrivilege=TRUE;
370          $PrivilegeTemplate.QualifierFormats = #propertyvalues[#i];
371       }
372       else {
373          $Role.< #propertynames[#i]> = #propertyvalues[#i];
374          #atLeastOneRole = TRUE;
375       }
376    }
377    if (#atLeastOnePrivilege) {
378    //#PrivilegeTemplateString contains the embedded instance produced from
379       $PrivilegeTemplate using conversion mechanism presumed to exist
380    #PrivilegeTemplateStrings[0] = $PrivilegeTemplateString;
381    %InArguments[] = { newArgument("Role", $Role->),
382                      newArgument("Privileges", #PrivilegeTemplateStrings[]->) }
383    %OutArguments[] = {  };
384    //invoke method
385    #returnStatus = smOpInvokeMethod ($SvcInstancePaths[0].GetObjectPath(),
386       "ModifyRole",
387       %InArguments[],
388       %OutArguments[]);
389    //3 process return code to CLP Command Status
390    if (0 != #Error.code)  {
391       //method invocation failed
392       if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )    {
393          // if the method invocation contains an embedded error
394          // use it for the Error for the overall job
395          &smAddError($job, #Error.$error[0]);
396          &smMakeCommandStatus($job);
397          &smEnd;
398       }
399       else if (#Error.code == 17)   {
400       //trap for CIM_METHOD_NOT_FOUND
```

```
401      //and make nice Unsupported msg.
402         //unsupported
403         $OperationError = smNewInstance("CIM_Error");
404         //CIM_ERR_NOT_SUPPORTED
405         $OperationError.CIMStatusCode = 7;
406         //Other
407         $OperationError.ErrorType = 1;
408         //Low
409         $OperationError.PerceivedSeverity = 2;
410         $OperationError.OwningEntity = DMTF:SMCLP;
411         $OperationError.MessageID = 0x00000001;
412         $OperationError.Message = "Operation is not supported.";
413         &smAddError($job, $OperationError);
414         &smMakeCommandStatus($job);
415         &smEnd;
416      }
417      else if (#Error.code != 0) {
418         //operation failed, but no detailed error instance, need to make one up
419         //make an Error instance and associate with job for Operation
420            $OperationError = smNewInstance("CIM_Error");
421            //CIM_ERR_FAILED
422            $OperationError.CIMStatusCode = 1;
423            //Software Error
424            $OperationError.ErrorType = 4;
425            //Unknown
426            $OperationError.PerceivedSeverity = 0;
427            $OperationError.OwningEntity = DMTF:SMCLP;
428            $OperationError.MessageID = 0x00000009;
429            $OperationError.Message = "An internal software error has occurred.";
430            &smAddError($job, $OperationError);
431            &smMakeCommandStatus($job);
432            &smEnd;
433      }
434 }//if CIM op failed
435 }//at least one privilege
436 if (#atLeastOneRole)   {
437    #Error = &smOpModifyInstance ( $Role, NULL );
438    if (0 != #Error.code) {
439        &smProcessOpError (#Error);
440        //includes &smEnd;
441    }
442 }//at least one Role
443 //modify worked, show the resultant role
444 &lShowRole($Role->, #all);
```

6   Mappings

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

6.1   CIM_ConcreteDependency

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_ConcreteDependency

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

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

6.1.1   Ordering of Results

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

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

6.1.2   Show

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

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

469 **6.1.2.1   Show Multiple Instances – CIM_Role Reference**

470 This command form is for the `show` verb applied to multiple instances. This command form corresponds
471 to a `show` command issued against CIM_ConcreteDependency where only one reference is specified and
472 the reference is to an instance of CIM_Role. An instance of CIM_Role is referenced by at most one
473 instance of CIM_ConcreteDependency

474 **6.1.2.1.1   Command Form**

475 `show <CIM_ConcreteDependency single instance>`

476 **6.1.2.1.2   CIM Requirements**

477 See CIM_ConcreteDependency in the "CIM Elements" section of the *Role Based Authorization Profile* for
478 the list of mandatory properties.

479 **6.1.2.1.3   Behavior Requirements**

480 **6.1.2.1.3.1   Preconditions**

481 `$instance` contains the instance of CIM_Role that is referenced by CIM_ConcreteDependency.

482 **6.1.2.1.3.2   Pseudo Code**

483 `&smShowAssociationInstances ( "CIM_ConcreteDependency", $instance.getObjectPath() );`
484 `&smEnd;`

485 **6.1.2.2   Show Multiple Instances – CIM_Privilege Reference**

486 This command form is for the `show` verb applied to multiple instances. This command form corresponds
487 to a `show` command issued against CIM_ConcreteDependency where only one reference is specified and
488 the reference is to an instance of CIM_Privilege.

489 **6.1.2.2.1   Command Form**

490 `show <CIM_ConcreteDependency multiple instances>`

491 **6.1.2.2.2   CIM Requirements**

492 See CIM_ConcreteDependency in the "CIM Elements" section of the *Role Based Authorization Profile* for
493 the list of mandatory properties.

494 **6.1.2.2.3   Behavior Requirements**

495 **6.1.2.2.3.1   Preconditions**

496 `$instance` contains the instance of CIM_Privilege that is referenced by CIM_ConcreteDependency.

497 **6.1.2.2.3.2   Pseudo Code**

498 `&smShowAssociationInstances ( "CIM_ConcreteDependency", $instance.getObjectPath() );`
499 `&smEnd;`

500 **6.1.2.3   Show Multiple Instances – CIM_RoleBasedAuthorizationService Reference**

501 This command form is for the `show` verb applied to multiple instances. This command form corresponds
502 to a `show` command issued against CIM_ConcreteDependency where only one reference is specified and
503 the reference is to an instance of CIM_RoleBasedAuthorizationService.

6.1.2.3.1   Command Form

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

6.1.2.3.2   CIM Requirements

See CIM_ConcreteDependency in the "CIM Elements" section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.1.2.3.3   Behavior Requirements

6.1.2.3.3.1   Preconditions

$instance contains the instance of CIM_RoleBasedAuthorizationService that is referenced by CIM_ConcreteDependency.

6.1.2.3.3.2   Pseudo Code

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

6.1.2.4   Show a Single Instance – CIM_Identity Reference

This command form is for the show verb applied to a single instance. This command form corresponds to the show command issued against CIM_ConcreteDependency where the reference specified is to an instance of CIM_Identity. The *Role Based Authorization Profile* requires that an instance of CIM_Identity be referenced by zero or one instance of CIM_ConcreteDependency. Therefore, at most a single instance will be returned.

6.1.2.4.1   Command Form

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

6.1.2.4.2   CIM Requirements

See CIM_ConcreteDependency in the "CIM Elements" section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.1.2.4.3   Behavior Requirements

6.1.2.4.3.1   Preconditions

$instance contains the instance of CIM_Identity which is referenced by CIM_ConcreteDependency.

6.1.2.4.3.2   Pseudo Code

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

6.1.2.5   Show a Single Instance – CIM_Role and CIM_Identity References

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

537 **6.1.2.5.1   Command Form**

538 `show <CIM_ConcreteDependency single instance>`

539 **6.1.2.5.2   CIM Requirements**

540 See CIM_ConcreteDependency in the "CIM Elements" section of the *Role Based Authorization Profile* for
541 the list of mandatory properties.

542 **6.1.2.5.3   Behavior Requirements**

543 **6.1.2.5.3.1   Preconditions**

544 `$instanceA` contains the instance of CIM_Identity which is referenced by CIM_ConcreteDependency.

545 `$instanceB` contains the instance of CIM_Role which is referenced by CIM_ConcreteDependency.

546 **6.1.2.5.3.2   Pseudo Code**

547 `&smShowAssociationInstance ( "CIM_ConcreteDependency", $instanceA.getObjectPath(),`
548 `    $instanceB.getObjectPath() );`
549 `&smEnd;`

550 **6.1.2.6   Show a Single Instance – CIM_Privilege and CIM_RoleBasedAuthorizationService**
551 **References**

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

555 **6.1.2.6.1   Command Form**

556 `show <CIM_ConcreteDependency single instance>`

557 **6.1.2.6.2   CIM Requirements**

558 See CIM_ConcreteDependency in the "CIM Elements" section of the *Role Based Authorization Profile* for
559 the list of mandatory properties.

560 **6.1.2.6.3   Behavior Requirements**

561 **6.1.2.6.3.1   Preconditions**

562 `$instanceA` contains the instance of CIM_Privilege which is referenced by CIM_ConcreteDependency.

563 `$instanceB` contains the instance of CIM_RoleBasedAuthorizationService which is referenced by
564 CIM_ConcreteDependency.

565 **6.1.2.6.3.2   Pseudo Code**

566 `&smShowAssociationInstance ( "CIM_ConcreteDependency", $instanceA.getObjectPath(),`
567 `    $instanceB.getObjectPath() );`
568 `&smEnd;`

569 ## 6.2   CIM_ElementCapabilities

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

571 Table 3 lists each SM CLP verb, the required level of support for the verb in conjunction with instances of
572 the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the
573 verb and target. Table 3 is for informational purposes only; in case of a conflict between Table 3 and
574 requirements detailed in the following sections, the text detailed in the following sections supersedes the
575 information in Table 3.

576   **Table 3 – 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	

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

6.2.1   Ordering of Results

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

582   • Results for CIM_ElementCapabilities are unordered; therefore, no algorithm is defined.

6.2.2   Show

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

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

6.2.2.1   Show Multiple Instances – CIM_RoleBasedManagementCapabilities Reference

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

6.2.2.1.1   Command Form

594 `show <CIM_ElementCapabilities multiple instances>`

6.2.2.1.2   CIM Requirements

596 See CIM_ElementCapabilities in the "CIM Elements" section of the *Role Based Authorization Profile* for
597 the list of mandatory properties.

598   **6.2.2.1.3   Behavior Requirements**

599   **6.2.2.1.3.1   Preconditions**

600   $instance contains the instance of CIM_RoleBasedManagementCapabilities which is referenced by
601   CIM_ElementCapabilities.

602   **6.2.2.1.3.2   Pseudo Code**

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

605   **6.2.2.2   Show a Single Instance – CIM_RoleBasedAuthorizationService Reference**

606   This command form is for the show verb applied to a single instance. This command form corresponds to
607   the show command issued against CIM_ElementCapabilities where the reference specified is to an
608   instance of CIM_Role.

609   **6.2.2.2.1   Command Form**

610   **show <CIM_ElementCapabilities *single instance*>**

611   **6.2.2.2.2   CIM Requirements**

612   See CIM_ElementCapabilities in the "CIM Elements" section of the *Role Based Authorization Profile* for
613   the list of mandatory properties.

614   **6.2.2.2.3   Behavior Requirements**

615   **6.2.2.2.3.1   Preconditions**

616   $instance contains the instance of CIM_RoleBasedAuthorizationService which is referenced by
617   CIM_ElementCapabilities.

618   **6.2.2.2.3.2   Pseudo Code**

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

621   **6.2.2.3   Show a Single Instance – CIM_RoleBasedManagementCapabilities and**
622   **CIM_RoleBasedAuthorizationService References**

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

626   **6.2.2.3.1   Command Form**

627   **show <CIM_ElementCapabilities *single instance*>**

628   **6.2.2.3.2   CIM Requirements**

629   See CIM_ElementCapabilities in the "CIM Elements" section of the *Role Based Authorization Profile* for
630   the list of mandatory properties.

631   **6.2.2.3.3   Behavior Requirements**

632   **6.2.2.3.3.1   Preconditions**

633   `$instanceA` contains the instance of CIM_RoleBasedManagementCapabilities which is referenced by
634   CIM_ElementCapabilities.

635   `$instanceB` contains the instance of CIM_RoleBasedAuthorizationService which is referenced by
636   CIM_ElementCapabilities.

637   **6.2.2.3.3.2   Pseudo Code**

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

641   **6.3   CIM_HostedService**

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

643   Table 4 lists each SM CLP verb, the required level of support for the verb in conjunction with instances of
644   the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the
645   verb and target. Table 4 is for informational purposes only; in case of a conflict between Table 4 and
646   requirements detailed in the following sections, the text detailed in the following sections supersedes the
647   information in Table 4.

648   **Table 4 – 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 6.3.2.
start	Not supported	
stop	Not supported	

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

651   **6.3.1   Ordering of Results**

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

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

655 **6.3.2 Show**

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

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

659 **6.3.2.1 Show Multiple Instances – CIM_ComputerSystem Reference**

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

663 **6.3.2.1.1 Command Form**

664 `show <CIM_HostedService multiple instances>`

665 **6.3.2.1.2 CIM Requirements**

666 See CIM_HostedService in the "CIM Elements" section of the *Role Based Authorization Profile* for the list
667 of mandatory properties.

668 **6.3.2.1.3 Behavior Requirements**

669 **6.3.2.1.3.1 Preconditions**

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

671 **6.3.2.1.3.2 Pseudo Code**

672 `&smShowAssociationInstances ( "CIM_HostedService", $instance.getObjectPath() );`
673 `&smEnd;`

674 **6.3.2.2 Show a Single Instance – CIM_RoleBasedAuthorizationService Reference**

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

679 **6.3.2.2.1 Command Form**

680 `show <CIM_HostedService single instance>`

681 **6.3.2.2.2 CIM Requirements**

682 See CIM_HostedService in the "CIM Elements" section of the *Role Based Authorization Profile* for the list
683 of mandatory properties.

684 **6.3.2.2.3 Behavior Requirements**

685 **6.3.2.2.3.1 Preconditions**

686 `$instance` contains the instance of CIM_RoleBasedAuthorizationService which is referenced by
687 CIM_HostedService.

6.3.2.2.3.2 Pseudo Code

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

6.3.2.3 Show a Single Instance – Both References

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

6.3.2.3.1 Command Form

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

6.3.2.3.2 CIM Requirements

See CIM_HostedService in the "CIM Elements" section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.3.2.3.3 Behavior Requirements

6.3.2.3.3.1 Preconditions

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

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

6.3.2.3.3.2 Pseudo Code

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

6.4   CIM_MemberOfCollection

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

Table 5 lists each SM CLP verb, the required level of support for the verb in conjunction with the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and target. Table 5 is for informational purposes only; in case of a conflict between Table 5 and requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 5.

Table 5 – Command Verb Requirements for CIM_MemberOfCollection

Command Verb	Requirement	Comments
create	May	See 6.4.2.
delete	May	See 6.4.3.
dump	Not supported	None
load	Not supported	None
reset	Not supported	None
set	Not supported	None

Command Verb	Requirement	Comments
show	Shall	See 6.4.4.
start	Not supported	None
stop	Not supported	None

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

6.4.1   Ordering of Results

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

723   • Results for CIM_MemberOfCollection are unordered; therefore, no algorithm is defined.

6.4.2   Create

725 This section describes how to implement the `create` verb when applied to an instance of
726 CIM_MemberOfCollection. Implementations may support the use of the `create` verb with
727 CIM_MemberOfCollection.

728 The `create` verb is used to create an instance of CIM_MemberOfCollection that associates an instance
729 of CIM_Identity with an instance of CIM_Role.

6.4.2.1   Create Specifying Both References

731 In order to create an instance of CIM_MemberOfCollection, a client is required to supply references to
732 both ManagedElements.

6.4.2.1.1   Command Form

734 ```
create <reference 1> <CIM_MemberOfCollection> <reference 2>
```

#### 6.4.2.1.2   CIM Requirements

736 See CIM_MemberOfCollection in the "CIM Elements" section of the *Role Based Authorization Profile* for
737 the list of mandatory properties.

#### 6.4.2.1.3   Behavior Requirements

#### 6.4.2.1.3.1   Preconditions

740 ```
//reference to the CIM_Role instance or the CIM_Identity Instance
```
741 ```
$ref1 = <reference 1>;
```
742 ```
//reference to the CIM_Identity instance when $ref1 is the CIM_Role instance or the
```
743 ```
   CIM_Role instance if $ref1 is the CIM_Identity instance
```
744 ```
$ref2 = <reference 2>;
```
745 ```
$assoc = smNewInstance ("CIM_MemberOfCollection");
```
746 ```
// validate parameters, this may be done by CreateInstance too, but the level of error
```
747 ```
     detection and validation is not documented in 1.0 profile
```
748 ```
if ($ref1 instanceof CIM_Role)    {
```
749 ```
   if (!($ref2 instanceof CIM_Identity))   {
```
750 ```
   $OperationError = smNewInstance("CIM_Error");
```
751 ```
   //CIM_ERR_FAILED
```

```
752      $OperationError.CIMStatusCode = 1;
753      //Software Error
754      $OperationError.ErrorType = 4;
755      //Unknown
756      $OperationError.PerceivedSeverity = 0;
757      $OperationError.OwningEntity = DMTF:SMCLP;
758      $OperationError.MessageID = 0x00000010;
759      $OperationError.Message = "The target association can not be created between the
760          specified targets.";
761      &smAddError($job, $OperationError);
762      &smMakeCommandStatus($job);
763      &smEnd;
764      }
765      else   {
766          $assoc.GroupComponent = $ref1.ObjectPath();
767          $assoc.PartComponent = $ref2.ObjectPath();
768          &smCreateInstance($assoc);
769      }
770  }
771  else if ($ref1 instanceof CIM_Identity)  {
772      if ( !($ref2 instanceof CIM_Role))    {
773          $OperationError = smNewInstance("CIM_Error");
774          //CIM_ERR_FAILED
775          $OperationError.CIMStatusCode = 1;
776          //Software Error
777          $OperationError.ErrorType = 4;
778          //Unknown
779          $OperationError.PerceivedSeverity = 0;
780          $OperationError.OwningEntity = DMTF:SMCLP;
781          $OperationError.MessageID = 0x00000010;
782          $OperationError.Message = "The target association can not be created between
783              the specified targets.";
784              &smAddError($job, $OperationError);
785              &smMakeCommandStatus($job);
786              &smEnd;
787      }
788      else   {
789          $assoc.GroupComponent = $ref2.ObjectPath();
790          $assoc.PartComponent = $ref1.ObjectPath();
791          &smCreateInstance($assoc);
792      }
793  }
794  else   {
795      $OperationError = smNewInstance("CIM_Error");
796      //CIM_ERR_FAILED
797      $OperationError.CIMStatusCode = 1;
798      //Software Error
799      $OperationError.ErrorType = 4;
800      //Unknown
```

```
801     $OperationError.PerceivedSeverity = 0;
802     $OperationError.OwningEntity = DMTF:SMCLP;
803     $OperationError.MessageID = 0x00000010;
804     $OperationError.Message = "The target association can not be created between the
805         specified targets.";
806     &smAddError($job, $OperationError);
807     &smMakeCommandStatus($job);
808     &smEnd;
809   }
```

6.4.3   Delete a Single Instance

811   Delete a single instance of CIM_MemberOfCollection.

6.4.3.1   Command Form

813   **delete <CIM_MemberOfCollection *single instance*>**

6.4.3.2   CIM Requirements

815   See CIM_MemberOfCollection in the "CIM Elements" section of the *Role Based Authorization Profile* for
816   the list of mandatory properties.

6.4.3.3   Behavior Requirements

```
818   $instance=<CIM_MemberOfCollection single instance>
819   &smDeleteInstance ( $instance.GetObjectPath() );
```

6.4.4   Show

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

6.4.4.1   Show Multiple Instances – CIM_Identity Reference

825   This command form is for the show command applied to CIM_MemberOfCollection where only the
826   reference to an instance of CIM_Identity is specified. An instance of CIM_Identity can be referenced by
827   zero or more instances of CIM_Identity.

6.4.4.1.1   Command Form

829   **show <CIM_MemberOfCollection *multiple instances*>**

6.4.4.1.2   CIM Requirements

831   See CIM_MemberOfCollection in the "CIM Elements" section of the *Role Based Authorization Profile* for
832   the list of mandatory properties.

6.4.4.1.3   Behavior Requirements

6.4.4.1.3.1   Preconditions

835   $instance contains the instance of CIM_Idenity that is referenced by CIM_MemberOfCollection.

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

6.4.4.1.3.2   Pseudo Code

```
#propertylist[] = NULL;
if ( false == #all)
    {
    #propertylist[] = {//all mandatory non-key properties};
    }
&smShowAssociationInstances ( "CIM_MemberOfCollection", $instance.getObjectPath(),
    #propertylist[] );
&smEnd;
```

6.4.4.2   Show Multiple Instances – CIM_Role Reference

This command form is for the show command applied to CIM_MemberOfCollection where only the reference to an instance of CIM_Role is specified. An instance of CIM_Role can be referenced by zero or more instances of CIM_MemberOfCollection.

6.4.4.2.1   Command Form

```
show <CIM_MemberOfCollection single instances>
```

6.4.4.2.2   CIM Requirements

See CIM_MemberOfCollection in the "CIM Elements" section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.4.4.2.3   Behavior Requirements

6.4.4.2.3.1   Preconditions

$instance contains the instance of CIM_Role that is referenced by CIM_MemberOfCollection.

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

6.4.4.2.3.2   Pseudo Code

```
#propertylist[] = NULL;
if ( false == #all)
    {
    #propertylist[] = {//all mandatory non-key properties};
    }
&smShowAssociationInstances ( "CIM_MemberOfCollection", $instance.getObjectPath(),
    #propertylist[] );
&smEnd;
```

6.4.4.3   Show a Single Instance – Both References

This command form is for the show command applied to CIM_MemberOfCollection where both references are specified. Therefore, exactly one instance is shown.

6.4.4.3.1   Command Form

```
show <CIM_MemberOfCollection single instance>
```

873 **6.4.4.3.2 CIM Requirements**

874 See CIM_MemberOfCollection in the "CIM Elements" section of the *Role Based Authorization Profile* for
875 the list of mandatory properties.

876 **6.4.4.3.3 Behavior Requirements**

877 **6.4.4.3.3.1 Preconditions**

878 `$instanceA` contains one of the instances of CIM_Role referenced by CIM_MemberOfCollection.

879 `$instanceB` contains one of the instances of CIM_Identity referenced by CIM_MemberOfCollection.

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

881 **6.4.4.3.3.2 Pseudo Code**

```
882  #propertylist[] = NULL;
883  if ( false == #all)
884     {
885     #propertylist[] = {//all mandatory non-key properties};
886     }
887  &smShowAssociationInstance ( "CIM_MemberOfCollection", $instanceA.getObjectPath(),
888     $instanceB.getObjectPath(), #propertylist[] );
889  &smEnd;
```

890 **6.5 CIM_OwningCollectionElement**

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

892 Table 6 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
893 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
894 target. Table 6 is for informational purposes only; in case of a conflict between Table 6 and requirements
895 detailed in the following sections, the text detailed in the following sections supersedes the information in
896 Table 6.

897 **Table 6 – Command Verb Requirements for CIM_OwningCollectionElement**

Command Verb	Requirement	Comments
create	Not supported	None
delete	Not supported	None
dump	Not supported	None
load	Not supported	None
reset	Not supported	None
set	Not supported	None
show	Shall	See 6.5.2.
start	Not supported	None
stop	Not supported	None

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

6.5.1 Ordering of Results

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

- Results for CIM_OwningCollectionElement 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_OwningCollectionElement. Implementations shall support the use of the `show` verb with CIM_OwningCollectionElement.

6.5.2.1 Show a Single Instance – Both References

This command form is for the `show` command applied to CIM_OwningCollectionElement where both references are specified. Therefore, a single instance will be returned.

6.5.2.1.1 Command Form

```
show <CIM_OwningCollectionElement single instance>
```

6.5.2.1.2 CIM Requirements

See CIM_OwningCollectionElement in the "CIM Elements" section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.5.2.1.3 Behavior Requirements

6.5.2.1.3.1 Preconditions

`$instanceA` contains one of the instances of CIM_Role that is referenced by CIM_OwningCollectionElement.

`$instanceB` contains one of the instances of <CIM_ComputerSystem> that is referenced by CIM_OwningCollectionElement.

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

6.5.2.1.3.2 Pseudo Code

```
#propertylist[] = NULL;
if ( false == #all)
    {
    #propertylist[] = {//all mandatory non-key properties};
    }
&smShowAssociationInstance ( "CIM_OwningCollectionElement",
    $instanceA.getObjectPath(), $instanceB.getObjectPath(), #propertylist[] );
&smEnd;
```

6.5.2.2 Show Multiple Instances – CIM_ComputerSystem Reference

This command form is for the `show` command applied to CIM_OwningCollectionElement where only the reference to an instance of CIM_ComputerSystem is specified. An instance of CIM_ComputerSystem can be referenced by multiple instances of CIM_OwningCollectionElement.

936 **6.5.2.2.1 Command Form**

937 `show <CIM_OwningCollectionElement multiple instances>`

938 **6.5.2.2.2 CIM Requirements**

939 See CIM_OwningCollectionElement in the "CIM Elements" section of the *Role Based Authorization Profile*
940 for the list of mandatory properties.

941 **6.5.2.2.3 Behavior Requirements**

942 **6.5.2.2.3.1 Preconditions**

943 `$instance` contains the instance of CIM_ComputerSystem that is referenced by
944 CIM_OwningCollectionElement.

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

946 **6.5.2.2.3.2 Pseudo Code**

```
947  #propertylist[] = NULL;
948  if ( false == #all)
949      {
950      #propertylist[] = {//all mandatory non-key properties};
951      }
952  &smShowAssociationInstances ( "CIM_OwningCollectionElement",
953      $instance.getObjectPath(), #propertylist[] );
954  &smEnd;
```

955 **6.5.2.3 Show a Single Instance – CIM_Role reference**

956 This command form is for the `show` command applied to CIM_OwningCollectionElement where only the
957 reference to an instance of CIM_Role is specified. An instance of CIM_Role is referenced by exactly one
958 instance of CIM_OwningCollectionElement. Therefore, a single instance is returned.

959 **6.5.2.3.1 Command Form**

960 `show <CIM_OwningCollectionElement single instances>`

961 **6.5.2.3.2 CIM Requirements**

962 See CIM_OwningCollectionElement in the "CIM Elements" section of the *Role Based Authorization Profile*
963 for the list of mandatory properties.

964 **6.5.2.3.3 Behavior Requirements**

965 **6.5.2.3.3.1 Preconditions**

966 `$instance` contains the instance of <CIM_Role> that is referenced by CIM_OwningCollectionElement.

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

968 **6.5.2.3.3.2 Pseudo Code**

```
969   #propertylist[] = NULL;
970   if ( false == #all)
971       {
972       #propertylist[] = {//all mandatory non-key properties};
973       }
974   &smShowAssociationInstances ( "CIM_OwningCollectionElement",
975       $instance.getObjectPath(), #propertylist[] );
976   &smEnd;
```

6.6 CIM_Privilege

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 the target
980 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
981 target. Table 7 is for informational purposes only; in case of a conflict between Table 7 and requirements
982 detailed in the following sections, the text detailed in the following sections supersedes the information in
983 Table 7.

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

Command Verb	Requirement	Comments
create	Not supported	None
delete	Not supported	None
dump	Not supported	None
load	Not supported	None
reset	Not supported	None
set	May	See 6.6.2.
show	Shall	See 6.6.3.
start	Not supported	None
stop	Not supported	None

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

6.6.1 Ordering of Results

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

990 • Results for CIM_Privilege are unordered; therefore, no algorithm is defined.

6.6.2 Set

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

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

6.6.2.1 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 *Role Based Authorization Profile*.

6.6.2.1.1 Command Form

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

6.6.2.1.2 CIM Requirements

See CIM_Privilege in the "CIM Elements" section of the *Role Based Authorization Profile* for the list of modifiable properties.

6.6.2.1.3 Behavior Requirements

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

6.6.2.2 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 *Role Based Authorization Profile*.

6.6.2.2.1 Command Form

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

6.6.2.2.2 CIM Requirements

See CIM_Privilege in the "CIM Elements" section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.6.2.2.3 Behavior Requirements

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

6.6.3   Show

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

6.6.3.1   Show a Single Instance

Show a single instance of CIM_Privilege.

6.6.3.1.1   Command Form

```
show <CIM_Privilege single instance>
```

6.6.3.1.2   CIM Requirements

See CIM_Privilege in the "CIM Elements" section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.6.3.1.3   Behavior Requirements

6.6.3.1.3.1   Preconditions

`$instance` represents the targeted instance of CIM_Privilege.

```
$instance=<CIM_Privilege single instance>
```

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

6.6.3.1.3.2   Pseudo Code

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

6.6.3.2   Show Multiple Instances

Show multiple instances of CIM_Privilege when they are addressed through an instance of CIM_RoleBasedAuthorizationService.

6.6.3.2.1   Command Form

```
show <CIM_Privilege multiple instances>
```

6.6.3.2.2   CIM Requirements

See CIM_Privilege in the "CIM Elements" section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.6.3.2.3   Behavior Requirements

6.6.3.2.3.1   Preconditions

`$containerInstance` contains the instance of CIM_RoleBasedAuthorizationService for which we are displaying related CIM_Privilege instances.

1070   `#all` is true if the "-all" option was specified with the command; otherwise, `#all` is false.

1071   **6.6.3.2.3.2   Pseudo Code**

```
1072    #propertylist[] = NULL;
1073    if (false == #all)
1074        {
1075        #propertylist[] = { //all mandatory non-key properties };
1076        }
1077    &smShowInstances ( "CIM_Privilege", "CIM_ConcreteDependency",
1078        $containerInstance.getObjectPath(), #propertylist[] );
1079    &smEnd;
```

1080   ## 6.7   CIM_Role

1081   The `cd` and `help` verbs shall be supported as described in DSP0216.

1082   Table 8 lists each SM CLP verb, the required level of support for the verb in conjunction with instances of
1083   the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the
1084   verb and target. Table 8 is for informational purposes only; in case of a conflict between Table 8 and
1085   requirements detailed in the following sections, the text detailed in the following sections supersedes the
1086   information in Table 8.

1087   **Table 8 – Command Verb Requirements for CIM_Role**

Command Verb	Requirement	Comments
create	May	See 6.7.2.
delete	May	See 6.7.3.
dump	Not supported	
load	Not supported	
reset	Not supported	
set	May	See 6.7.4.
show	Shall	See 6.7.5.
start	Not supported	
stop	Not supported	

1088   No mapping is defined for the following verbs for the specified target: `dump`, `load`, `reset`, `start`, and
1089   `stop`.

1090   ### 6.7.1   Ordering of Results

1091   When results are returned for multiple instances of CIM_Role, implementations shall utilize the following
1092   algorithm to produce the natural (that is, default) ordering:

1093   • Results for CIM_Role are unordered; therefore, no algorithm is defined.

1094   ### 6.7.2   Create

1095   This section describes how to implement the `create` verb when applied to an instance of CIM_Role.
1096   Implementations may support the use of the `create` verb with CIM_Role.

1097   The create verb is used to create an additional Role on the system.

1098   **6.7.2.1   Create Specifying Properties, Privileges and Scope of Role**

1099   **6.7.2.1.1   Command Form**

1100   **create <CIM_Role *single instance*> <CIM_Role propertyname1>=<propertyvalue1>**
1101   **<CIM_Role propertyname*n*>=<propertyvalue*n*>**

1102   **6.7.2.1.2   CIM Requirements**

1103   See CIM_Role in the "CIM Elements" section of the *Role Based Authorization Profile* for
1104   CIM_RoleBasedAuthorizationService.CreateRole.

1105   **6.7.2.1.3   Behavior Requirements**

1106   **6.7.2.1.3.1   Preconditions**

1107 $system contains the instance of CIM_ComputerSystem that is the Resultant Target of the CLP
1108   command. This is the container instance for the desired Role instance.

1109   **6.7.2.1.3.2   Pseudo Code**

```
1110    //1 Find the Service for the system to create the Role
1111    #Error = &smOpAssociators(
1112            $system.getObjectPath(),
1113            "CIM_HostedService",
1114            "CIM_RoleBasedAuthorizationService",
1115            NULL,
1116            NULL,
1117            NULL,
1118            $SvcInstancePaths[])
1119    if (0 != #Error.code)
1120    {
1121            &smProcessOpError (#Error);
1122            //includes &smEnd;
1123    }
1124    //no service associated
1125    if (NULL == $ SvcInstancePaths[0]) {
1126       $OperationError = smNewInstance("CIM_Error");
1127       //CIM_ERR_FAILED
1128       $OperationError.CIMStatusCode = 1;
1129       //Software Error
1130       $OperationError.ErrorType = 4;
1131       //Unknown
1132       $OperationError.PerceivedSeverity = 0;
1133       $OperationError.OwningEntity = DMTF:SMCLP;
1134       $OperationError.MessageID = 0x00000001;
1135       $OperationError.Message = "Operation is not supported";
1136       &smAddError($job, $OperationError);
1137       &smMakeCommandStatus($job);
1138       &smEnd;
1139    }
```

```
1140    $service = $SvcInstancePaths[0];

1141    //2 convert command line parms to method parms
1142    $RoleTemplate = smNewInstance ("CIM_Role");
1143    $PrivilegeTemplate = smNewInstance ("CIM_Privilege");
1144    $PrivilegeTemplate.PrivilegesGranted = TRUE;
1145    for #i < n
1146    {
1147        if (<propertname#i> == "Activities") {
1148            $PrivilegeTemplate.Activities = <propertyvalue#i>;
1149        }
1150        else if (<propertname#i> == "ActivityQualifiers")    {
1151        $PrivilegeTemplate.ActivityQualifiers = <propertyvalue#i>;
1152        }
1153        else if (<propertname#i> == "QualifierFormats") {
1154        $PrivilegeTemplate.QualifierFormats = <propertyvalue#i>;
1155        }
1156        else {
1157            $RoleTemplate.<propertname#i> = $RoleTemplate.<propertyvalue#i>;
1158        }
1159    }
1160    //#RoleTemplateString contains the embedded instance produced from $RoleTemplate
1161    //using conversion mechanism presumed to exist
1162    //#PrivilegeTemplateString contains the embedded instance produced from
1163    //$PrivilegeTemplate using conversion mechanism presumed to exist
1164    #PrivilegeTemplateStrings[0] = $PrivilegeTemplateString;

1165
1166    $Role = smNewInstance ("CIM_Role");
1167    //build the parameter lists and invoke the method
1168    %InArguments[] = {newArgument("RoleTemplate", $RoleTemplateString),
1169                    newArgument("OwningSystem", $system.GetObjectPath()),
1170                    newArgument("Privileges", #PrivilegeTemplateStrings[])};
1171    %OutArguments[] = { newArgument("Role",
1172                        $Role.GetObjectPath()) };
1173    //invoke method
1174    #returnStatus = smOpInvokeMethod ($Service.GetObjectPath(),
1175        "CreateRole",
1176        %InArguments[],
1177        %OutArguments[]);
1178    //3 process return code to CLP Command Status
1179    if (0 != #Error.code) {
1180        //method invocation failed
1181        if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )    {
1182            // if the method invocation contains an embedded error
1183            // use it for the Error for the overall job
1184            &smAddError($job, #Error.$error[0]);
1185            &smMakeCommandStatus($job);
1186            &smEnd;
1187        }
```

```
1188      else if (#Error.code == 17)   {
1189      //trap for CIM_METHOD_NOT_FOUND
1190      //and make nice Unsupported msg.
1191         //unsupported
1192         $OperationError = smNewInstance("CIM_Error");
1193         //CIM_ERR_NOT_SUPPORTED
1194         $OperationError.CIMStatusCode = 7;
1195         //Other
1196         $OperationError.ErrorType = 1;
1197         //Low
1198         $OperationError.PerceivedSeverity = 2;
1199         $OperationError.OwningEntity = DMTF:SMCLP;
1200         $OperationError.MessageID = 0x00000001;
1201         $OperationError.Message = "Operation is not supported.";
1202         &smAddError($job, $OperationError);
1203         &smMakeCommandStatus($job);
1204         &smEnd;
1205      }
1206      else {
1207         //operation failed, but no detailed error instance, need to make one up
1208         //make an Error instance and associate with job for Operation
1209            $OperationError = smNewInstance("CIM_Error");
1210            //CIM_ERR_FAILED
1211            $OperationError.CIMStatusCode = 1;
1212            //Software Error
1213            $OperationError.ErrorType = 4;
1214            //Unknown
1215            $OperationError.PerceivedSeverity = 0;
1216            $OperationError.OwningEntity = DMTF:SMCLP;
1217            $OperationError.MessageID = 0x00000009;
1218            $OperationError.Message = "An internal software error has occurred.";
1219            &smAddError($job, $OperationError);
1220            &smMakeCommandStatus($job);
1221            &smEnd;
1222      }
1223 }//if CIM op failed
1224 else if (0 == #returnStatus) {
1225      //completed successfully, show created instance
1226      &lShowRole($Role, #all);
1227      &smEnd;
1228 }
1229 else if (4 == #returnStatus) {
1230      //generic failure
1231      $OperationError = smNewInstance("CIM_Error");
1232      //CIM_ERR_FAILED
1233      $OperationError.CIMStatusCode = 1;
1234      //Other
1235      $OperationError.ErrorType = 1;
1236      //Low
```

```
1237        $OperationError.PerceivedSeverity = 2;
1238        $OperationError.OwningEntity = DMTF:SMCLP;
1239        $OperationError.MessageID = 0x00000002;
1240        $OperationError.Message = "Failed.  No further information is available.";
1241        &smAddError($job, $OperationError);
1242        &smMakeCommandStatus($job);
1243 }
1244 else   {
1245        //invalid parameter
1246        $OperationError = smNewInstance("CIM_Error");
1247        //CIM_ERR_FAILED
1248        $OperationError.CIMStatusCode = 1;
1249        //Other
1250        $OperationError.ErrorType = 1;
1251        //Low
1252        $OperationError.PerceivedSeverity = 2;
1253        $OperationError.OwningEntity = DMTF:SMCLP;
1254        $OperationError.MessageID = 0x00000004;
1255        $OperationError.Message = "One or more parameters specified are invalid.";
1256        &smAddError($job, $OperationError);
1257        &smMakeCommandStatus($job);
1258        &smEnd;
1259 }
```

6.7.3   Delete

1260

1261 This section describes how to implement the delete verb when applied to an instance of CIM_Role.
1262 Implementations may support the use of the delete verb with CIM_Role.

1263 The delete command is used to remove an instance of CIM_Role.

6.7.3.1   Delete a Single Instance

1264

1265 Delete a single instance of CIM_Role.

6.7.3.1.1   Command Form

1266

1267 **delete <CIM_Role *single instance*>**

6.7.3.1.2   CIM Requirements

1268

1269 See CIM_Role in the "CIM Elements" section of the *Role Based Authorization Profile* for
1270 CIM_RoleBasedAuthorizationService.DeleteRole().

6.7.3.1.3   Behavior Requirements

1271

```
1272 //$Role-> contains the Object Path of the targeted role instance
1273 //1 Find the Service for the Role
1274 #Error = &smOpAssociators(
1275          $Role->,
1276          "CIM_ServiceAffectsElement",
1277          "CIM_RoleBasedAuthorizationService",
1278          NULL,
```

```
1279            NULL,
1280            NULL,
1281            $SvcInstancePaths[])
1282    if (0 != #Error.code)
1283    {
1284            &smProcessOpError (#Error);
1285            //includes &smEnd;
1286    }
1287    $service = $SvcInstancePaths[0];
1288    //build the parameter lists and invoke the method
1289    %InArguments[] = {newArgument("Role", $Role->)};
1290    %OutArguments[] = { };
1291    //invoke method
1292    #returnStatus = smOpInvokeMethod ($Service.GetObjectPath(),
1293       "DeleteRole",
1294       %InArguments[],
1295       %OutArguments[]);
1296    //3 process return code to CLP Command Status
1297    if (0 != #Error.code)  {
1298       //method invocation failed
1299       if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )    {
1300          // if the method invocation contains an embedded error
1301          // use it for the Error for the overall job
1302          &smAddError($job, #Error.$error[0]);
1303          &smMakeCommandStatus($job);
1304          &smEnd;
1305       }
1306       else if (#Error.code == 17)   {
1307       //trap for CIM_METHOD_NOT_FOUND
1308       //and make nice Unsupported msg.
1309          //unsupported
1310          $OperationError = smNewInstance("CIM_Error");
1311          //CIM_ERR_NOT_SUPPORTED
1312          $OperationError.CIMStatusCode = 7;
1313          //Other
1314          $OperationError.ErrorType = 1;
1315          //Low
1316          $OperationError.PerceivedSeverity = 2;
1317          $OperationError.OwningEntity = DMTF:SMCLP;
1318          $OperationError.MessageID = 0x00000001;
1319          $OperationError.Message = "Operation is not supported.";
1320          &smAddError($job, $OperationError);
1321          &smMakeCommandStatus($job);
1322          &smEnd;
1323       }
1324       else {
1325          //operation failed, but no detailed error instance, need to make one up
1326          //make an Error instance and associate with job for Operation
1327             $OperationError = smNewInstance("CIM_Error");
```

```
1328            //CIM_ERR_FAILED
1329            $OperationError.CIMStatusCode = 1;
1330            //Software Error
1331            $OperationError.ErrorType = 4;
1332            //Unknown
1333            $OperationError.PerceivedSeverity = 0;
1334            $OperationError.OwningEntity = DMTF:SMCLP;
1335            $OperationError.MessageID = 0x00000009;
1336            $OperationError.Message = "An internal software error has occurred.";
1337            &smAddError($job, $OperationError);
1338            &smMakeCommandStatus($job);
1339            &smEnd;
1340      }
1341 }//if CIM op failed
1342 else {
1343      //completed successfully, show created instance
1344      &smCommandCompleted($job);
1345      &smEnd;
1346 }
```

6.7.4   Set

1347

1348 This section describes how to implement the set verb when it is applied to an instance of CIM_Role.
1349 Implementations may support the use of the set verb with CIM_Role.

1350 The set verb is used to modify descriptive properties of the CIM_Role instance.

6.7.4.1   General Usage of Set for a Single Property

1351

1352 This command form corresponds to the general usage of the set verb to modify a single property of a
1353 target instance. This is the most common case.

1354 The requirement for supporting modification of a property using this command form shall be equivalent to
1355 the requirement for supporting modification of the property using the ModifyInstance operation as defined
1356 in the *Role Based Authorization Profile*.

6.7.4.1.1   Command Form

1357

```
1358 set <CIM_Role single instance> <propertyname>=<propertyvalue>
```

6.7.4.1.2   CIM Requirements

1359

1360 See CIM_Role in the "CIM Elements" section of the *Role Based Authorization Profile* for the list of
1361 modifiable properties.

6.7.4.1.3   Behavior Requirements

1362

```
1363 $instance=<CIM_Role single instance>
1364 #propertyNames[] = {<propertyname>};
1365 #propertyValues[] = {<propertyvalue>};
1366 &lModifyRole ( $instance, #propertyNames[], #propertyValues[], #all );
1367 &smEnd;
```

6.7.4.2 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 *Role Based Authorization Profile*.

6.7.4.2.1 Command Form

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

6.7.4.2.2 CIM Requirements

See CIM_Role in the "CIM Elements" section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.7.4.2.3 Behavior Requirements

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

6.7.5 Show

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

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

6.7.5.1 Show a Single Instance

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

6.7.5.1.1 Command Form

```
show <CIM_Role single instance>
```

6.7.5.1.2 CIM Requirements

See CIM_Role in the "CIM Elements" section of the *Role Based Authorization Profile* for the list of mandatory properties.

1402 **6.7.5.1.3 Behavior Requirements**

1403 **6.7.5.1.3.1 Preconditions**

1404 $instance represents the targeted instance of CIM_Role.

1405 `$instance=<CIM_Role single instance>`

1406 #all is true if the "-all" option was specified with the command; otherwise, #all is false.

1407 **6.7.5.1.3.2 Pseudo Code**

```
1408  #propertylist[] = NULL;
1409  if (false == #all)
1410      {
1411      #propertylist[] = { //all mandatory non-key properties };
1412      }
1413  &lShowRole ( $instance.getObjectPath(), #propertylist[] );
1414  &smEnd;
```

1415 **6.7.5.2 Show Multiple Instances**

1416 This command form is for the `show` verb applied to multiple instances of CIM_Role. This command form
1417 corresponds to UFsT-based selection within a scoping system.

1418 **6.7.5.2.1 Command Form**

1419 **`show <CIM_Role multiple instances>`**

1420 **6.7.5.2.2 CIM Requirements**

1421 See CIM_Role in the "CIM Elements" section of the *Role Based Authorization Profile* for the list of
1422 mandatory properties.

1423 **6.7.5.2.3 Behavior Requirements**

1424 **6.7.5.2.3.1 Preconditions**

1425 $containerInstance contains the instance of CIM_ComputerSystem for which we are displaying
1426 scoped Roles (CIM_Role instances). The *Role Based Authorization Profile* requires that the CIM_Role
1427 instance be associated with its scoping system via an instance of the CIM_OwningCollectionElement
1428 association.

1429 #all is true if the "-all" option was specified with the command; otherwise, #all is false.

1430 **6.7.5.2.3.2 Pseudo Code**

```
1431  #propertylist[] = NULL;
1432  if (false == #all)
1433      {
1434      #propertylist[] = { //all mandatory non-key properties };
1435      }
1436  //fetch all of the Roles
1437      #Error = &smOpAssociators ( $containerInstancePath->,
1438          "CIM_OwningCollectionElement", "CIM_Role",  NULL, NULL, NULL,
1439          $outInstancePaths->[] );
```

```
1440      if (0 != #Error.code)
1441      {
1442      &smProcessOpError (#Error);
1443      //includes &smEnd;
1444      }
1445      for #i in $outInstancePaths->[]
1446      {
1447      &lShowRole($outInstancePaths->[i]);
1448      }
1449 &smEnd;
```

6.8   CIM_RoleBasedManagementCapabilities

1451   The cd and help verbs shall be supported as described in DSP0216.

1452   Table 9 lists each SM CLP verb, the required level of support for the verb in conjunction with instances of
1453   the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the
1454   verb and target. Table 9 is for informational purposes only; in case of a conflict between Table 9 and
1455   requirements detailed in the following sections, the text detailed in the following sections supersedes the
1456   information in Table 9.

1457   **Table 9 – Command Verb Requirements for CIM_RoleBasedManagementCapabilities**

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

1458   No mapping is defined for the following verbs for the specified target: create, delete, dump, load,
1459   reset, set, start, and stop.

6.8.1   Ordering of Results

1461   When results are returned for multiple instances of CIM_RoleBasedManagementCapabilities,
1462   implementations shall utilize the following algorithm to produce the natural (that is, default) ordering:

1463   • Results for CIM_RoleBasedManagementCapabilities are unordered; therefore, no algorithm is
1464   defined.

6.8.2   Show

1466   This section describes how to implement the show verb when applied to an instance of
1467   CIM_RoleBasedManagementCapabilities. Implementations shall support the use of the show verb with
1468   CIM_RoleBasedManagementCapabilities.

1469 The show verb is used to display information about an instance or instances of the
1470 CIM_RoleBasedManagementCapabilities class.

6.8.2.1 Show a Single Instance

1472 This command form is for the show verb applied to a single instance of
1473 CIM_RoleBasedManagementCapabilities.

6.8.2.1.1 Command Form

1475 `show <CIM_RoleBasedManagementCapabilities single instance>`

6.8.2.1.2 CIM Requirements

1477 See CIM_RoleBasedManagementCapabilities in the "CIM Elements" section of the *Role Based*
1478 *Authorization Profile* for the list of mandatory properties.

6.8.2.1.3 Behavior Requirements

6.8.2.1.3.1 Preconditions

1481 $instance represents the targeted instance of CIM_RoleBasedManagementCapabilities.

1482 `$instance=<CIM_RoleBasedManagementCapabilities single instance>`

1483 #all is true if the "-all" option was specified with the command; otherwise, #all is false.

6.8.2.1.3.2 Pseudo Code

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

6.8.2.2 Show Multiple Instances

1493 This command form is for the show verb applied to multiple instances of
1494 CIM_RoleBasedManagementCapabilities. This command form corresponds to UFsT-based selection
1495 within a capabilities collection.

6.8.2.2.1 Command Form

1497 `show <CIM_RoleBasedManagementCapabilities multiple instances>`

6.8.2.2.2 CIM Requirements

1499 See CIM_RoleBasedManagementCapabilities in the "CIM Elements" section of the *Role Based*
1500 *Authorization Profile* for the list of mandatory properties.

6.8.2.2.3 Behavior Requirements

6.8.2.2.3.1 Preconditions

1503 $containerInstance represents the instance of CIM_ConcreteCollection with ElementName property
1504 that contains "Capabilities" and is associated with the targeted instances of
1505 CIM_RoleBasedManagementCapabilities through the CIM_MemberOfCollection association.

1506 `#all` is true if the "-all" option was specified with the command; otherwise, `#all` is false.

1507 **6.8.2.2.3.2 Pseudo Code**

```
1508 #propertylist[] = NULL;
1509 if ( false == #all )
1510     {
1511     #propertylist[] = { //all mandatory non-key properties }
1512     }
1513 &smShowInstances ( "CIM_RoleBasedManagementCapabilities", "CIM_MemberOfCollection",
1514     $containerInstance.getObjectPath(), #propertylist[] );
1515 &smEnd;
```

1516 ## 6.9   CIM_RoleBasedAuthorizationService

1517 The `cd` and `help` verbs shall be supported as described in <u>DSP0216</u>.

1518 Table 10 lists each SM CLP verb, the required level of support for the verb in conjunction with instances
1519 of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the
1520 verb and target. Table 10 is for informational purposes only; in case of a conflict between Table 10 and
1521 requirements detailed in the following sections, the text detailed in the following sections supersedes the
1522 information in Table 10.

1523 **Table 10 – Command Verb Requirements for CIM_RoleBasedAuthorizationService**

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 6.9.2.
start	Not supported	
stop	Not supported	

1524 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,
1525 `reset`, `set`, `start`, and `stop`.

1526 **6.9.1   Ordering of Results**

1527 When results are returned for multiple instances of CIM_RoleBasedAuthorizationService,
1528 implementations shall utilize the following algorithm to produce the natural (that is, default) ordering:

1529 • Results for CIM_RoleBasedAuthorizationService are unordered; therefore, no algorithm is
1530 defined.

1531 **6.9.2   Show**

1532 This section describes how to implement the `show` verb when applied to an instance of
1533 CIM_RoleBasedAuthorizationService. Implementations shall support the use of the `show` verb with
1534 CIM_RoleBasedAuthorizationService.

1535   The `show` verb is used to display information about the CIM_RoleBasedAuthorizationService.

1536   **6.9.2.1   Show a Single Instance**

1537   This command form is for the `show` verb applied to a single instance of
1538   CIM_RoleBasedAuthorizationService.

1539   **6.9.2.1.1   Command Form**

1540   `show <CIM_RoleBasedAuthorizationService single instance>`

1541   **6.9.2.1.2   CIM Requirements**

1542   See CIM_RoleBasedAuthorizationService in the "CIM Elements" section of the *Role Based Authorization*
1543   *Profile* for the list of mandatory properties.

1544   **6.9.2.1.3   Behavior Requirements**

1545   **6.9.2.1.3.1   Preconditions**

1546   `$instance` represents the targeted instance of CIM_RoleBasedAuthorizationService.

1547   `$instance=<CIM_RoleBasedAuthorizationService single instance>`

1548   `#all` is true if the "-all" option was specified with the command; otherwise, `#all` is false.

1549   **6.9.2.1.3.2   Pseudo Code**

```
1550  #propertylist[] = NULL;
1551  if ( false == #all )
1552      {
1553      #propertylist[] = { "ElementName" };
1554      }
1555  &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
1556  &smEnd;
```

1557   **6.9.2.2   Show Multiple Instances**

1558   This command form is for the `show` verb applied to multiple instances of
1559   CIM_RoleBasedAuthorizationService. This command form corresponds to UFsT-based selection within a
1560   scoping system.

1561   **6.9.2.2.1   Command Form**

1562   `show <CIM_RoleBasedAuthorizationService multiple instances>`

1563   **6.9.2.2.2   CIM Requirements**

1564   See CIM_RoleBasedAuthorizationService in the "CIM Elements" section of the *Role Based Authorization*
1565   *Profile* for the list of mandatory properties.

1566   **6.9.2.2.3   Behavior Requirements**

1567   **6.9.2.2.3.1   Preconditions**

1568   `$containerInstance` contains the instance of CIM_ComputerSystem for which we are displaying
1569   scoped instances of the CIM_RoleBasedAuthorizationService. The *Role Based Authorization Profile*

1570   requires that the CIM_RoleBasedAuthorizationService instance be associated with its scoping system via
1571   an instance of the CIM_HostedService association.

1572   #all is true if the "-all" option was specified with the command; otherwise, #all is false.

1573   **6.9.2.2.3.2 Pseudo Code**

```
1574    #propertylist[] = NULL;
1575    if ( false == #all )
1576        {
1577        #propertylist[] = { "ElementName" };
1578        }
1579    &smShowInstances ( "CIM_RoleBasedAuthorizationService", "CIM_HostedService",
1580        $containerInstance.getObjectPath(), #propertylist[] );
1581    &smEnd;
```

6.10 CIM_RoleLimitedToTarget

1583   The cd and help verbs shall be supported as described in DSP0216.

1584   Table 11 lists each SM CLP verb, the required level of support for the verb in conjunction with instances
1585   of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the
1586   verb and target. Table 11 is for informational purposes only; in case of a conflict between Table 11 and
1587   requirements detailed in the following sections, the text detailed in the following sections supersedes the
1588   information in Table 11.

1589   **Table 11 – Command Verb Requirements for CIM_RoleLimitedToTarget**

Command Verb	Requirement	Comments
create	May	See 6.10.2.
delete	May	See 6.10.3.
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.10.4.
start	Not supported	
stop	Not supported	

1590   No mapping is defined for the following verbs for the specified target: dump, load, reset, set, start,
1591   and stop.

6.10.1 Ordering of Results

1593   When results are returned for multiple instances of CIM_RoleLimitedToTarget, implementations shall
1594   utilize the following algorithm to produce the natural (that is, default) ordering:

1595   • Results for CIM_RoleLimitedToTarget are unordered; therefore, no algorithm is defined.

6.10.2 Create

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

6.10.2.1 Create Specifying Both References

In order to create an instance of CIM_RoleLimitedToTarget, a client is required to supply references to an instance of CIM_Role and the targeted ManagedElement.

6.10.2.1.1 Command Form

```
create <reference 1> <CIM_RoleLimitedToTarget> <reference 2>
```

6.10.2.1.2 CIM Requirements

See CIM_RoleLimitedToTarget in the "CIM Elements" section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.10.2.1.3 Behavior Requirements

6.10.2.1.3.1 Steps

1) Determine target Role and target Managed Element (ME).

2) Find the associated instance of CIM_RoleBasedAuthorizationService.

3) Find all instances of CIM_RoleLimitedToTarget that reference target Role.

4) Find all instances of CIM_ManagedElement referenced by associations.

5) Add target ME to list.

6) Invoke ModifyRole() with array of MEs.

7) Show the created association.

6.10.2.1.3.2 Pseudo Code

```
// $instance1=<reference 1>
// $instance2=<reference 2>
$Role-> = NULL;
$ME-> = NULL;
if ($instance1-> instanceof CIM_Role){
      $Role-> = $instance1->;
      $ME-> = $instance2->;
}
else if ( $instance2-> instanceof CIM_Role ) {
      $Role-> = $instance2->;
      $ME-> = $instance2->;
}
else  {
   $OperationError = smNewInstance("CIM_Error");
   //CIM_ERR_FAILED
   $OperationError.CIMStatusCode = 1;
   //Software Error
   $OperationError.ErrorType = 4;
```

```
1636        //Unknown
1637        $OperationError.PerceivedSeverity = 0;
1638        $OperationError.OwningEntity = DMTF:SMCLP;
1639        $OperationError.MessageID = 0x00000010;
1640        $OperationError.Message = "The target association can not be created between the
1641            specified targets.";
1642        &smAddError($job, $OperationError);
1643        &smMakeCommandStatus($job);
1644        &smEnd;
1645    }
1646    #Error = &smOpAssociatorNames(
1647            $Role->,
1648            "CIM_RoleLimitedToTarget",
1649            "CIM_ManagedElement",
1650            NULL,
1651            NULL,
1652            NULL,
1653            $MEInstancePaths->[])
1654    if (0 != #Error.code)
1655    {
1656            &smProcessOpError (#Error);
1657            //includes &smEnd;
1658    }
1659    #MEInstancePaths->[#MEInstancePaths.length] = $ME->
1660    #Error = &smOpAssociators(
1661            $Role->,
1662            "CIM_ServiceAffectsElement",
1663            "CIM_RoleBasedAuthorizationService",
1664            NULL,
1665            NULL,
1666            NULL,
1667            $SvcInstancePaths[])
1668    if (0 != #Error.code)
1669    {
1670            &smProcessOpError (#Error);
1671            //includes &smEnd;
1672    }
1673    // invoke the method
1674    %InArguments[] = { newArgument("Role", $Role->),
1675    newArgument("RoleLimitedToTargets", #MEInstancePaths->[]) }
1676    %OutArguments[] = {   };
1677    //invoke method
1678    #returnStatus = smOpInvokeMethod ($SvcInstancePaths[0].GetObjectPath(),
1679        "ModifyRole",
1680        %InArguments[],
1681        %OutArguments[]);
1682    //3 process return code to CLP Command Status
1683    if (0 != #Error.code)  {
1684        //method invocation failed
```

```
1685     if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )     {
1686         // if the method invocation contains an embedded error
1687         // use it for the Error for the overall job
1688         &smAddError($job, #Error.$error[0]);
1689         &smMakeCommandStatus($job);
1690         &smEnd;
1691     }
1692     else if (#Error.code == 17)    {
1693     //trap for CIM_METHOD_NOT_FOUND
1694     //and make nice Unsupported msg.
1695         //unsupported
1696         $OperationError = smNewInstance("CIM_Error");
1697         //CIM_ERR_NOT_SUPPORTED
1698         $OperationError.CIMStatusCode = 7;
1699         //Other
1700         $OperationError.ErrorType = 1;
1701         //Low
1702         $OperationError.PerceivedSeverity = 2;
1703         $OperationError.OwningEntity = DMTF:SMCLP;
1704         $OperationError.MessageID = 0x00000001;
1705         $OperationError.Message = "Operation is not supported.";
1706         &smAddError($job, $OperationError);
1707         &smMakeCommandStatus($job);
1708         &smEnd;
1709     }
1710     else if (#Error.code != 0) {
1711         //operation failed, but no detailed error instance, need to make one up
1712         //make an Error instance and associate with job for Operation
1713         $OperationError = smNewInstance("CIM_Error");
1714         //CIM_ERR_FAILED
1715         $OperationError.CIMStatusCode = 1;
1716         //Software Error
1717         $OperationError.ErrorType = 4;
1718         //Unknown
1719         $OperationError.PerceivedSeverity = 0;
1720         $OperationError.OwningEntity = DMTF:SMCLP;
1721         $OperationError.MessageID = 0x00000009;
1722         $OperationError.Message = "An internal software error has occurred.";
1723         &smAddError($job, $OperationError);
1724         &smMakeCommandStatus($job);
1725         &smEnd;
1726     }
1727 }//if CIM op failed
1728 //modify worked, show the resultant association instance
1729     &smShowAssociationInstance ( "CIM_RoleLimitedToTarget", $instance1->,
1730         $instance2->);
1731     &smEnd;
1732 }
```

6.10.3 Delete

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

6.10.3.1 Delete Specifying Both References

In order to delete an instance of CIM_RoleLimitedToTarget, a client is required to supply references to an
instance of CIM_Role and the targeted ManagedElement.

6.10.3.1.1 Command Form

```
Delete <reference 1> <CIM_RoleLimitedToTarget> <reference 2>
```

6.10.3.1.2 CIM Requirements

See CIM_RoleLimitedToTarget in the "CIM Elements" section of the *Role Based Authorization Profile* for
the list of mandatory properties.

6.10.3.1.3 Behavior Requirements

6.10.3.1.3.1 Steps

    1)   Determine target Role and target ME.

    2)   Find the associated instance of CIM_RoleBasedAuthorizationService.

    3)   Find all instances of CIM_RoleLimitedToTarget that reference target Role.

    4)   Find all instances of CIM_ManagedElement referenced by associations.

    5)   Remove target ME from list.

    6)   Invoke ModifyRole() with array of MEs.

6.10.3.1.3.2 Pseudo Code

```
// $instance1=<reference 1>
// $instance2=<reference 2>
$Role-> = NULL;
$ME-> = NULL;
if ($instance1-> instanceof CIM_Role){
      $Role-> = $instance1->;
      $ME-> = $instance2->;
}
else if ( $instance2-> instanceof CIM_Role ) {
      $Role-> = $instance2->;
      $ME-> = $instance2->;
}
else  {
   $OperationError = smNewInstance("CIM_Error");
   //CIM_ERR_FAILED
   $OperationError.CIMStatusCode = 1;
   //Software Error
   $OperationError.ErrorType = 4;
   //Unknown
   $OperationError.PerceivedSeverity = 0;
```

```
1774        $OperationError.OwningEntity = DMTF:SMCLP;
1775        $OperationError.MessageID = 0x00000010;
1776        $OperationError.Message = "The target association can not be Deleted between the
1777            specified targets.";
1778        &smAddError($job, $OperationError);
1779        &smMakeCommandStatus($job);
1780        &smEnd;
1781    }
1782    #Error = &smOpAssociatorNames(
1783            $Role->,
1784            "CIM_RoleLimitedToTarget",
1785            "CIM_ManagedElement",
1786            NULL,
1787            NULL,
1788            NULL,
1789            $MEInstancePaths->[])
1790    if (0 != #Error.code)
1791    {
1792        &smProcessOpError (#Error);
1793        //includes &smEnd;
1794    }
1795    for (#i=0; #i<$MEInstancePaths->[].length; #i++) {
1796    if (#MEInstancePaths->[#i] == $ME->)  {
1797        #MEInstancePaths->[#i] = NULL;
1798    }
1799    #Error = &smOpAssociators(
1800            $Role->,
1801            "CIM_ServiceAffectsElement",
1802            "CIM_RoleBasedAuthorizationService",
1803            NULL,
1804            NULL,
1805            NULL,
1806            $SvcInstancePaths[])
1807    if (0 != #Error.code)
1808    {
1809        &smProcessOpError (#Error);
1810        //includes &smEnd;
1811    }
1812    // invoke the method
1813    %InArguments[] = { newArgument("Role", $Role->),
1814    newArgument("RoleLimitedToTargets", #MEInstancePaths->[]) }
1815    %OutArguments[] = {  };
1816    //invoke method
1817    #returnStatus = smOpInvokeMethod ($SvcInstancePaths[0].GetObjectPath(),
1818        "ModifyRole",
1819        %InArguments[],
1820        %OutArguments[]);
1821    //3 process return code to CLP Command Status
1822    if (0 != #Error.code)   {
```

```
1823        //method invocation failed
1824        if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) ) {
1825            // if the method invocation contains an embedded error
1826            // use it for the Error for the overall job
1827            &smAddError($job, #Error.$error[0]);
1828            &smMakeCommandStatus($job);
1829            &smEnd;
1830        }
1831        else if (#Error.code == 17) {
1832        //trap for CIM_METHOD_NOT_FOUND
1833        //and make nice Unsupported msg.
1834            //unsupported
1835            $OperationError = smNewInstance("CIM_Error");
1836            //CIM_ERR_NOT_SUPPORTED
1837            $OperationError.CIMStatusCode = 7;
1838            //Other
1839            $OperationError.ErrorType = 1;
1840            //Low
1841            $OperationError.PerceivedSeverity = 2;
1842            $OperationError.OwningEntity = DMTF:SMCLP;
1843            $OperationError.MessageID = 0x00000001;
1844            $OperationError.Message = "Operation is not supported.";
1845            &smAddError($job, $OperationError);
1846            &smMakeCommandStatus($job);
1847            &smEnd;
1848        }
1849        else if (#Error.code != 0) {
1850            //operation failed, but no detailed error instance, need to make one up
1851            //make an Error instance and associate with job for Operation
1852                $OperationError = smNewInstance("CIM_Error");
1853                //CIM_ERR_FAILED
1854                $OperationError.CIMStatusCode = 1;
1855                //Software Error
1856                $OperationError.ErrorType = 4;
1857                //Unknown
1858                $OperationError.PerceivedSeverity = 0;
1859                $OperationError.OwningEntity = DMTF:SMCLP;
1860                $OperationError.MessageID = 0x00000009;
1861                $OperationError.Message = "An internal software error has occurred.";
1862                &smAddError($job, $OperationError);
1863                &smMakeCommandStatus($job);
1864                &smEnd;
1865        }
1866    }//if CIM op failed
1867    //modify worked, show the resultant association instance
1868        &smShowAssociationInstance ( "CIM_RoleLimitedToTarget", $instance1->,
1869            $instance2->);
1870        &smEnd;
1871    }
```

1872 **6.10.4 Show**

1873 This section describes how to implement the show verb when applied to an instance of
1874 CIM_AssignedIdentity. Implementations shall support the use of the show verb with
1875 CIM_AssignedIdentity.

1876 The show command is used to display information about the CIM_RoleLimitedToTarget instance or
1877 instances.

1878 **6.10.4.1 Show Multiple Instances – CIM_Role Reference**

1879 This command form is for the show verb applied to multiple instances. This command form corresponds
1880 to a show command issued against CIM_RoleLimitedToTarget where only one reference is specified and
1881 the reference is to an instance of CIM_Role.

1882 **6.10.4.1.1 Command Form**

1883 `show <CIM_RoleLimitedToTarget multiple instances>`

1884 **6.10.4.1.2 CIM Requirements**

1885 See CIM_RoleLimitedToTarget in the "CIM Elements" section of the *Role Based Authorization Profile* for
1886 the list of mandatory properties.

1887 **6.10.4.1.3 Behavior Requirements**

1888 **6.10.4.1.3.1 Preconditions**

1889 $instance contains the instance of CIM_Role that is referenced by CIM_RoleLimitedToTarget.

1890 **6.10.4.1.3.2 Pseudo Code**

1891 `&smShowAssociationInstances ( "CIM_RoleLimitedToTarget", $instance.getObjectPath() );`
1892 `&smEnd;`

1893 **6.10.4.2 Show Multiple Instances – CIM_ManagedElement reference**

1894 This command form is for the show verb applied to multiple instances. This command form corresponds
1895 to a show command issued against CIM_RoleLimitedToTarget where only one reference is specified and
1896 the reference is to an instance of CIM_ManagedElement subclass.

1897 **6.10.4.2.1 Command Form**

1898 `show <CIM_RoleLimitedToTarget multiple instances>`

1899 **6.10.4.2.2 CIM Requirements**

1900 See CIM_RoleLimitedToTarget in the "CIM Elements" section of the *Role Based Authorization Profile* for
1901 the list of mandatory properties.

1902 **6.10.4.2.3 Behavior Requirements**

1903 **6.10.4.2.3.1 Preconditions**

1904 $instance contains the instance of CIM_ManagedElement subclass that is referenced by
1905 CIM_RoleLimitedToTarget.

6.10.4.2.3.2 Pseudo Code

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

6.10.4.3 Show a Single Instance – Both References

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

6.10.4.3.1 Command Form

show <CIM_RoleLimitedToTarget *single instance*>

6.10.4.3.2 CIM Requirements

See CIM_RoleLimitedToTarget in the "CIM Elements" section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.10.4.3.3 Behavior Requirements

6.10.4.3.3.1 Preconditions

`$instanceA` contains the instance of CIM_Identity which is referenced by CIM_RoleLimitedToTarget.

`$instanceB` contains the instance of CIM_UserContact, CIM_Group, or CIM_Role which is referenced by CIM_RoleLimitedToTarget.

6.10.4.3.3.2 Pseudo Code

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

6.11   CIM_ServiceAffectsElement

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_ServiceAffectsElement

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

Command Verb	Requirement	Comments
set	Not supported	
show	Shall	See 6.11.2.
start	Not supported	
stop	Not supported	

1935 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,
1936 `reset`, `set`, `start`, and `stop`.

6.11.1 Ordering of Results

1938 When results are returned for multiple instances of CIM_ServiceAffectsElement, implementations shall
1939 utilize the following algorithm to produce the natural (that is, default) ordering:

1940 • Results for CIM_ServiceAffectsElement are unordered; therefore, no algorithm is defined.

6.11.2 Show

1942 This section describes how to implement the `show` verb when applied to an instance of
1943 CIM_ServiceAffectsElement. Implementations shall support the use of the `show` verb with
1944 CIM_ServiceAffectsElement.

1945 The `show` command is used to display information about the CIM_ServiceAffectsElement instance or
1946 instances.

6.11.2.1 Show Multiple Instances

1948 This command form is for the `show` verb applied to multiple instances. This command form corresponds
1949 to a `show` command issued against CIM_ServiceAffectsElement where only one reference is specified
1950 and the reference is to an instance of CIM_ComputerSystem.

6.11.2.1.1 Command Form

1952 **show <CIM_ServiceAffectsElement *multiple instances*>**

6.11.2.1.2 CIM Requirements

1954 See CIM_ServiceAffectsElement in the "CIM Elements" section of the *Role Based Authorization Profile* for
1955 the list of mandatory properties.

6.11.2.1.3 Behavior Requirements

6.11.2.1.3.1 Preconditions

1958 `$instance` contains the instance of CIM_ComputerSystem which is referenced by
1959 CIM_ServiceAffectsElement.

6.11.2.1.3.2 Pseudo Code

```
1961  &smShowAssociationInstances ( "CIM_ServiceAffectsElement",
1962     $instance.getObjectPath() );
1963  &smEnd;
```

6.11.2.2 Show a Single Instance – CIM_Role

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

6.11.2.2.1 Command Form

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

6.11.2.2.2 CIM Requirements

See CIM_ServiceAffectsElement in the "CIM Elements" section of the *Role Based Authorization 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_Role which is referenced by CIM_ServiceAffectsElement.

6.11.2.2.3.2 Pseudo Code

```
&smShowAssociationInstances ( "CIM_ServiceAffectsElement",
    $instance.getObjectPath() );
&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
the `show` command issued against CIM_ServiceAffectsElement where both references are specified and
therefore the desired instance is unambiguously identified.

6.11.2.3.1 Command Form

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

6.11.2.3.2 CIM Requirements

See CIM_ServiceAffectsElement in the "CIM Elements" section of the *Role Based Authorization 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_RoleBasedAuthorizationService which is referenced by
CIM_ServiceAffectsElement.

`$instanceB` contains the instance of CIM_Role or CIM_Role that is referenced by
CIM_ServiceAffectsElement.

6.11.2.3.3.2 Pseudo Code

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

6.12 CIM_ServiceServiceDependency

2000

2001 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

2002 Table 13 lists each SM CLP verb, the required level of support for the verb in conjunction with instances
2003 of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the
2004 verb and target. Table 13 is for informational purposes only; in case of a conflict between Table 13 and
2005 requirements detailed in the following sections, the text detailed in the following sections supersedes the
2006 information in Table 13.

2007 **Table 13 – Command Verb Requirements for CIM_ServiceServiceDependency**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.12.2.
start	Not supported	
stop	Not supported	

2008 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,
2009 `reset`, `set`, `start`, and `stop`.

6.12.1 Ordering of Results

2010

2011 When results are returned for multiple instances of CIM_ServiceServiceDependency, implementations
2012 shall utilize the following algorithm to produce the natural (that is, default) ordering:

2013 • Results for CIM_ServiceServiceDependency are unordered; therefore, no algorithm is defined.

6.12.2 Show

2014

2015 This section describes how to implement the `show` verb when applied to an instance of
2016 CIM_ServiceServiceDependency. Implementations shall support the use of the `show` verb with
2017 CIM_ServiceServiceDependency.

2018 The `show` command is used to display information about the CIM_ServiceServiceDependency instance
2019 or instances.

6.12.2.1 Show Multiple Instances – CIM_RoleBasedAuthorizationService

2020

2021 This command form is for the `show` verb applied to multiple instances. This command form corresponds
2022 to a `show` command issued against CIM_ServiceServiceDependency where only one reference is
2023 specified and the reference is to an instance of CIM_RoleBasedAuthorizationService.

6.12.2.1.1 Command Form

```
show <CIM_ServiceServiceDependency multiple objects>
```

6.12.2.1.2 CIM Requirements

See CIM_ServiceServiceDependency in the "CIM Elements" section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.12.2.1.3 Behavior Requirements

6.12.2.1.3.1 Preconditions

$instance contains the instance of CIM_RoleBasedAuthorizationService.

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

6.12.2.1.3.2 Pseudo Code

```
#propertylist[] = NULL;
if (#all == false)
    {
    #propertylist[] =  { //all mandatory non-key properties };
    }
&smShowAssociationInstances ( "CIM_ServiceServiceDependency",
    $instance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.12.2.2 Show Multiple Instances – CIM_AccountManagementService

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

6.12.2.2.1 Command Form

```
show <CIM_ServiceServiceDependency multiple objects>
```

6.12.2.2.2 CIM Requirements

See CIM_ServiceServiceDependency in the "CIM Elements" section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.12.2.2.3 Behavior Requirements

6.12.2.2.3.1 Preconditions

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

6.12.2.2.3.2 Pseudo Code

```
#propertylist[] = NULL;
if (#all == false)
    {
    #propertylist[] =  { //all mandatory non-key properties };
    }
&smShowAssociationInstances ( "CIM_ServiceServiceDependency",
    $instance.getObjectPath(), #propertylist[] );
&smEnd;
```

2063 **6.12.2.3 Show a Single Instances – Both References**

2064 This command form is for the `show` verb applied to multiple instances. This command form corresponds
2065 to a `show` command issued against CIM_ServiceServiceDependency where a reference is specified to an
2066 instance of CIM_AccountManagementService and the other reference is to an instance of
2067 CIM_RoleBasedAuthorizationService.

2068 **6.12.2.3.1 Command Form**

2069 `show <CIM_ServiceServiceDependency multiple objects>`

2070 **6.12.2.3.2 CIM Requirements**

2071 See CIM_ServiceServiceDependency in the "CIM Elements" section of the *Role Based Authorization*
2072 *Profile* for the list of mandatory properties.

2073 **6.12.2.3.3 Behavior Requirements**

2074 **6.12.2.3.3.1 Preconditions**

2075 `$instanceA` contains the instance of CIM_RoleBasedAuthorizationService which is referenced by
2076 CIM_ServiceServiceDependency.

2077 `$instanceB` contains the instance of CIM_AccountManagementService that is referenced by
2078 CIM_ServiceServiceDependency.

2079 **6.12.2.3.3.2 Pseudo Code**

2080 `&smShowAssociationInstance ( "CIM_ServiceServiceDependency",`
2081 `    $instanceA.getObjectPath(), $instanceB.getObjectPath() );`
2082 `&smEnd;`

2083

2084
2085

ANNEX A
(informative)

2086
2087
2088

Change Log

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

2089