Document Number: DSP0830

Date: 2009-07-14

Version: 1.0.0

Role Based Authorization Profile SM CLP Command Mapping Specification

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

CONTENTS

Tables

75

Foreword

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

Conventions

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

Acknowledgements

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

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

Introduction

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

The target audience for this specification is implementers of the SM CLP support for the *Role Based 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

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

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

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

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

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

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

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

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

4 Symbols and Abbreviated Terms

The following symbols and abbreviations are used in this document.

4.1
CIM
Common Information Model

4.2
CLP
Command Line Protocol

4.3
DMTF
Distributed Management Task Force

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
lShowRole	Show an instance of CIM_Role.	See 5.1.
lModifyRole	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 lShowRole

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.

3) Invoke ShowRoles().
4) Match target CIM_Role instance to embedded instance returned.
5) Find embedded instance of CIM_Privilege at corresponding array position.
6) Add the relevant CIM_Privilege properties to the CIM_Role instance as propagated properties.
7) Display the CIM_Role instance.

5.1.2 Pseudo Code

```
// $Role-> contains the object path of the CIM_Role to display
// #all indicates whether the all option was specified
sub lShowRole($Role->, #all)  {
   #propertylist[] = NULL;
   //if we're not displaying all of the properties, provide a list
   if (false == #all)  {
      #propertylist[] = { //all mandatory non-key properties };
   }
   //get the instance
   #Error = &smGetInstance ( $Role->, $Role );
   if (0 != #Error.code)     {
             &smProcessOpError (#Error);
             //includes &smEnd;
   }
#Error = &smOpAssociators(
       $Role->,
       “CIM_ServiceAffectsElement”,
       “CIM_RoleBasedAuthorizationService”,
       NULL,
       NULL,
       NULL,
       $SvcInstancePaths[])

if (0 != #Error.code)
{
      &smProcessOpError (#Error);
      //includes &smEnd;
}
// invoke the method
//build the parameter lists and invoke the method
%InArguments[] = { }
%OutArguments[] = { newArgument(“Roles”, #RoleStrings[]),
                    newArgument(“Privileges”, #PrivilegeStrings[]->) };
//invoke method
#returnStatus = smOpInvokeMethod ($SvcInstancePaths[0].GetObjectPath(),
   “ShowRoles”,
   %InArguments[],
   %OutArguments[]);
//3 process return code to CLP Command Status
if (0 != #Error.code)  {
   //method invocation failed
   if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )    {
      // if the method invocation contains an embedded error
      // use it for the Error for the overall job
```

```
      &smAddError($job, #Error.$error[0]);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else if (#Error.code == 17)   {
   //trap for CIM_METHOD_NOT_FOUND
   //and make nice Unsupported msg.
      //unsupported
      $OperationError = smNewInstance("CIM_Error");
      //CIM_ERR_NOT_SUPPORTED
      $OperationError.CIMStatusCode = 7;
      //Other
      $OperationError.ErrorType = 1;
      //Low
      $OperationError.PerceivedSeverity = 2;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000001;
      $OperationError.Message = "Operation is not supported.";
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else if (#Error.code != 0) {
      //operation failed, but no detailed error instance, need to make one up
      //make an Error instance and associate with job for Operation
         $OperationError = smNewInstance("CIM_Error");
         //CIM_ERR_FAILED
         $OperationError.CIMStatusCode = 1;
         //Software Error
         $OperationError.ErrorType = 4;
         //Unknown
         $OperationError.PerceivedSeverity = 0;
         $OperationError.OwningEntity = DMTF:SMCLP;
         $OperationError.MessageID = 0x00000009;
         $OperationError.Message = "An internal software error has occurred.";
         &smAddError($job, $OperationError);
         &smMakeCommandStatus($job);
         &smEnd;
   }
}//if CIM op failed
   //$Roles[] contains instances of CIM_Role generated from the embedded instances
      contained in #RoleStrings[]
   //$Privileges contains instances of CIM_Privilege generated from the embedded
      instances contained in #PrivilegeStrings[]
   //placeholder for privilege instance if one is found.
   $Privilege = NULL;
   for (#i-0; i<$Roles[].length; i++ {
      if ($Roles[#i].CommonName = $Role.CommonName) {
         $Privilege = $Privileges[#i];
         break;
      }
   }
   #referencedproperties[] = NULL;
```

```
   if (NULL != $Privilege){
      $Role.Activities = $Privilege.Activities;
      $Role.ActivityQualifiers = $Privilege.ActivityQualifiers;
      $Role.QualifierFormats = $Privilege.QualifierFormats;
      #referencedProperties[] =  { “Activities”, “ActivityQualifiers”,
         “QualifierFormats” };
   }
   &smShowInstancePseudoProperties ( $Role.GetObjectPath(), #propertyList[],
      #referencedProperties[] );
   &smEnd;
} //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

```
// $Role-> contains the object path of the CIM_Role to modify
// #propertyNames[] contains array of property names specified on the command line
//#propertyValues[] contains array of property values assigned for each property name
// #all indicates whether the all option was specified
sub lModifyRole($Role->, #propertyNames[], #propertyValues[], #all){
   //get the instance
   #Error = &smGetInstance ( $Role->, $Role );
   if (0 != #Error.code)     {
             &smProcessOpError (#Error);
             //includes &smEnd;
   }
#Error = &smOpAssociators(
         $Role->,
         “CIM_ServiceAffectsElement”,
         “CIM_RoleBasedAuthorizationService”,
         NULL,
         NULL,
         NULL,
         $SvcInstancePaths[])
if (0 != #Error.code)
{
       &smProcessOpError (#Error);
       //includes &smEnd;
}
```

```
// invoke the method
//build the parameter lists and invoke the method
$PrivilegeTemplate = smNewInstance (“CIM_Privilege”);
$PrivilegeTemplate.PrivilegesGranted = TRUE;
#atLeastOnePrivilege = FALSE;
#atLeastOneRole = FALSE;
for (#i < #propertynames[].length)
{
   if (#propertynames[#i] == “Activities”)  {
      $PrivilegeTemplate.Activities = #propertyvalues[#i];
      #atLeastOnePrivilege=TRUE;
   }
   else if (#propertynames[#i] == “ActivityQualifiers”) {
   $PrivilegeTemplate.ActivityQualifiers = #propertyvalues[#i];
      #atLeastOnePrivilege=TRUE;
   }
   else if (#propertynames[#i] == “QualifierFormats”)  {
      #atLeastOnePrivilege=TRUE;
      $PrivilegeTemplate.QualifierFormats = #propertyvalues[#i];
   }
   else {
      $Role.< #propertynames[#i]> = #propertyvalues[#i];
      #atLeastOneRole = TRUE;
   }
}
if (#atLeastOnePrivilege)  {
//#PrivilegeTemplateString contains the embedded instance produced from
   $PrivilegeTemplate using conversion mechanism presumed to exist
#PrivilegeTemplateStrings[0] = $PrivilegeTemplateString;
%InArguments[] = { newArgument(“Role”, $Role->),
                   newArgument(“Privileges”, #PrivilegeTemplateStrings[]->) }
%OutArguments[] = {  };
//invoke method
#returnStatus = smOpInvokeMethod ($SvcInstancePaths[0].GetObjectPath(),
   “ModifyRole”,
   %InArguments[],
   %OutArguments[]);
//3 process return code to CLP Command Status
if (0 != #Error.code)  {
   //method invocation failed
   if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )    {
      // if the method invocation contains an embedded error
      // use it for the Error for the overall job
      &smAddError($job, #Error.$error[0]);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else if (#Error.code == 17)   {
   //trap for CIM_METHOD_NOT_FOUND
```

```
   //and make nice Unsupported msg.
      //unsupported
      $OperationError = smNewInstance(“CIM_Error”);
      //CIM_ERR_NOT_SUPPORTED
      $OperationError.CIMStatusCode = 7;
      //Other
      $OperationError.ErrorType = 1;
      //Low
      $OperationError.PerceivedSeverity = 2;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000001;
      $OperationError.Message = “Operation is not supported.”;
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else if (#Error.code != 0) {
      //operation failed, but no detailed error instance, need to make one up
      //make an Error instance and associate with job for Operation
         $OperationError = smNewInstance(“CIM_Error”);
         //CIM_ERR_FAILED
         $OperationError.CIMStatusCode = 1;
         //Software Error
         $OperationError.ErrorType = 4;
         //Unknown
         $OperationError.PerceivedSeverity = 0;
         $OperationError.OwningEntity = DMTF:SMCLP;
         $OperationError.MessageID = 0x00000009;
         $OperationError.Message = “An internal software error has occurred.”;
         &smAddError($job, $OperationError);
         &smMakeCommandStatus($job);
         &smEnd;
   }
}//if CIM op failed
}//at least one privilege
if (#atLeastOneRole)   {
   #Error = &smOpModifyInstance ( $Role, NULL );
   if (0 != #Error.code) {
      &smProcessOpError (#Error);
      //includes &smEnd;
   }
}//at least one Role
//modify worked, show the resultant role
&lShowRole($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.

6.1.2.1 Show Multiple Instances – CIM_Role Reference

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

6.1.2.1.1 Command Form

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

6.1.2.1.2 CIM Requirements

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

6.1.2.1.3 Behavior Requirements

6.1.2.1.3.1 Preconditions

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

6.1.2.1.3.2 Pseudo Code

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

6.1.2.2 Show Multiple Instances – CIM_Privilege Reference

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

6.1.2.2.1 Command Form

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

6.1.2.2.2 CIM Requirements

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

6.1.2.2.3 Behavior Requirements

6.1.2.2.3.1 Preconditions

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

6.1.2.2.3.2 Pseudo Code

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

6.1.2.3 Show Multiple Instances – CIM_RoleBasedAuthorizationService Reference

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

6.1.2.5.1 Command Form

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

6.1.2.5.2 CIM Requirements

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

6.1.2.5.3 Behavior Requirements

6.1.2.5.3.1 Preconditions

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

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

6.1.2.5.3.2 Pseudo Code

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

6.1.2.6 Show a Single Instance – CIM_Privilege and CIM_RoleBasedAuthorizationService 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.

6.1.2.6.1 Command Form

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

6.1.2.6.2 CIM Requirements

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

6.1.2.6.3 Behavior Requirements

6.1.2.6.3.1 Preconditions

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

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

6.1.2.6.3.2 Pseudo Code

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

6.2 CIM_ElementCapabilities

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

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

Table 3 – Command Verb Requirements for CIM_ElementCapabilities

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.2.2.
start	Not supported	
stop	Not supported	

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

6.2.1 Ordering of Results

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

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

6.2.2 Show

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

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

6.2.2.1 Show Multiple Instances – CIM_RoleBasedManagementCapabilities Reference

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

6.2.2.1.1 Command Form

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

6.2.2.1.2 CIM Requirements

See CIM_ElementCapabilities in the “CIM Elements” section of the *Role Based Authorization Profile* for 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.

6.2.2.3.3 Behavior Requirements

6.2.2.3.3.1 Preconditions

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

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

6.2.2.3.3.2 Pseudo Code

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

6.3 CIM_HostedService

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

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

Table 4 – Command Verb Requirements for CIM_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	

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

6.3.1 Ordering of Results

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

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

6.3.2 Show

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

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

6.3.2.1 Show Multiple Instances – CIM_ComputerSystem Reference

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

6.3.2.1.1 Command Form

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

6.3.2.1.2 CIM Requirements

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

6.3.2.1.3 Behavior Requirements

6.3.2.1.3.1 Preconditions

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

6.3.2.1.3.2 Pseudo Code

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

6.3.2.2 Show a Single Instance – CIM_RoleBasedAuthorizationService 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_HostedService where the reference specified is to an instance of CIM_RoleBasedAuthorizationService. An instance of CIM_RoleBasedAuthorizationService is referenced by exactly one instance of CIM_HostedService. Therefore, a single instance will be returned.

6.3.2.2.1 Command Form

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

6.3.2.2.2 CIM Requirements

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

6.3.2.2.3 Behavior Requirements

6.3.2.2.3.1 Preconditions

`$instance` contains the instance of CIM_RoleBasedAuthorizationService which is referenced by 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

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

6.4.1 Ordering of Results

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

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

6.4.2 Create

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

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

6.4.2.1 Create Specifying Both References

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

6.4.2.1.1 Command Form

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

6.4.2.1.2 CIM Requirements

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

6.4.2.1.3 Behavior Requirements

6.4.2.1.3.1 Preconditions

```
//reference to the CIM_Role instance or the CIM_Identity Instance
$ref1 = <reference 1>;
//reference to the CIM_Identity instance when $ref1 is the CIM_Role instance or the
   CIM_Role instance if $ref1 is the CIM_Identity instance
$ref2 = <reference 2>;
$assoc = smNewInstance (“CIM_MemberOfCollection”);
// validate parameters, this may be done by CreateInstance too, but the level of error
      detection and validation is not documented in 1.0 profile
if ($ref1 instanceof CIM_Role)    {
   if ( !($ref2 instanceof CIM_Identity))    {
   $OperationError = smNewInstance(“CIM_Error”);
   //CIM_ERR_FAILED
```

```
   $OperationError.CIMStatusCode = 1;
   //Software Error
   $OperationError.ErrorType = 4;
   //Unknown
   $OperationError.PerceivedSeverity = 0;
   $OperationError.OwningEntity = DMTF:SMCLP;
   $OperationError.MessageID = 0x00000010;
   $OperationError.Message = “The target association can not be created between the
      specified targets.”;
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
   &smEnd;
   }
   else   {
      $assoc.GroupComponent = $ref1.ObjectPath();
      $assoc.PartComponent = $ref2.ObjectPath();
      &smCreateInstance($assoc);
   }
}
else if ($ref1 instanceof CIM_Identity)  {
   if ( !($ref2 instanceof CIM_Role))    {
      $OperationError = smNewInstance(“CIM_Error”);
      //CIM_ERR_FAILED
      $OperationError.CIMStatusCode = 1;
      //Software Error
      $OperationError.ErrorType = 4;
      //Unknown
      $OperationError.PerceivedSeverity = 0;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000010;
      $OperationError.Message = “The target association can not be created between
         the specified targets.”;
         &smAddError($job, $OperationError);
         &smMakeCommandStatus($job);
         &smEnd;
   }
   else   {
      $assoc.GroupComponent = $ref2.ObjectPath();
      $assoc.PartComponent = $ref1.ObjectPath();
      &smCreateInstance($assoc);
   }
}
else   {
   $OperationError = smNewInstance(“CIM_Error”);
   //CIM_ERR_FAILED
   $OperationError.CIMStatusCode = 1;
   //Software Error
   $OperationError.ErrorType = 4;
   //Unknown
```

```
    $OperationError.PerceivedSeverity = 0;
    $OperationError.OwningEntity = DMTF:SMCLP;
    $OperationError.MessageID = 0x00000010;
    $OperationError.Message = “The target association can not be created between the
       specified targets.”;
    &smAddError($job, $OperationError);
    &smMakeCommandStatus($job);
    &smEnd;
}
```

6.4.3 Delete a Single Instance

Delete a single instance of CIM_MemberOfCollection.

6.4.3.1 Command Form

```
delete <CIM_MemberOfCollection single instance>
```

6.4.3.2 CIM Requirements

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

6.4.3.3 Behavior Requirements

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

6.4.4 Show

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

6.4.4.1 Show Multiple Instances – CIM_Identity Reference

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

6.4.4.1.1 Command Form

```
show <CIM_MemberOfCollection multiple instances>
```

6.4.4.1.2 CIM Requirements

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

6.4.4.1.3 Behavior Requirements

6.4.4.1.3.1 Preconditions

`$instance` contains the instance of CIM_Idenitity that is referenced by CIM_MemberOfCollection.

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

6.4.4.1.3.2 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>
```

6.4.4.3.2 CIM Requirements

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

6.4.4.3.3 Behavior Requirements

6.4.4.3.3.1 Preconditions

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

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

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

6.4.4.3.3.2 Pseudo Code

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

6.5 CIM_OwningCollectionElement

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

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

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

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

6.5.1 Ordering of Results

When results are returned for multiple instances of CIM_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.

6.5.2.2.1 Command Form

```
show <CIM_OwningCollectionElement multiple instances>
```

6.5.2.2.2 CIM Requirements

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

6.5.2.2.3 Behavior Requirements

6.5.2.2.3.1 Preconditions

`$instance` contains the instance 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.

6.5.2.2.3.2 Pseudo Code

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

6.5.2.3 Show a Single Instance – CIM_Role reference

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

6.5.2.3.1 Command Form

```
show <CIM_OwningCollectionElement single instances>
```

6.5.2.3.2 CIM Requirements

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

6.5.2.3.3 Behavior Requirements

6.5.2.3.3.1 Preconditions

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

`#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;
```

977 ## 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`.

987 ### 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.

991 ### 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>
   <propertyname n>=<propertyvalue n>
```

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.

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

6.6.3.2.3.2 Pseudo Code

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

6.7 CIM_Role

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

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

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	

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

6.7.1 Ordering of Results

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

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

6.7.2 Create

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

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

6.7.2.1 Create Specifying Properties, Privileges and Scope of Role

6.7.2.1.1 Command Form

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

6.7.2.1.2 CIM Requirements

See CIM_Role in the “CIM Elements” section of the *Role Based Authorization Profile* for CIM_RoleBasedAuthorizationService.CreateRole.

6.7.2.1.3 Behavior Requirements

6.7.2.1.3.1 Preconditions

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

6.7.2.1.3.2 Pseudo Code

```
//1 Find the Service for the system to create the Role
#Error = &smOpAssociators(
       $system.getObjectPath(),
       “CIM_HostedService”,
       “CIM_RoleBasedAuthorizationService”,
       NULL,
       NULL,
       NULL,
       $SvcInstancePaths[])
if (0 != #Error.code)
{
       &smProcessOpError (#Error);
       //includes &smEnd;
}
//no service associated
if (NULL == $ SvcInstancePaths[0]) {
   $OperationError = smNewInstance(“CIM_Error”);
   //CIM_ERR_FAILED
   $OperationError.CIMStatusCode = 1;
   //Software Error
   $OperationError.ErrorType = 4;
   //Unknown
   $OperationError.PerceivedSeverity = 0;
   $OperationError.OwningEntity = DMTF:SMCLP;
   $OperationError.MessageID = 0x00000001;
   $OperationError.Message = “Operation is not supported”;
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
   &smEnd;
}
```

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

//2 convert command line parms to method parms
$RoleTemplate = smNewInstance (“CIM_Role”);
$PrivilegeTemplate = smNewInstance (“CIM_Privilege”);
$PrivilegeTemplate.PrivilegesGranted = TRUE;
for #i < n
{
   if (<propertname#i> == “Activities”)  {
      $PrivilegeTemplate.Activities = <propertyvalue#i>;
   }
   else if (<propertname#i> == “ActivityQualifiers”)    {
   $PrivilegeTemplate.ActivityQualifiers = <propertyvalue#i>;
   }
   else if (<propertname#i> == “QualifierFormats”)  {
   $PrivilegeTemplate.QualifierFormats = <propertyvalue#i>;
   }
   else {
      $RoleTemplate.<propertname#i> = $RoleTemplate.<propertyvalue#i>;
   }
}
//#RoleTemplateString contains the embedded instance produced from $RoleTemplate
//using conversion mechanism presumed to exist
//#PrivilegeTemplateString contains the embedded instance produced from
//$PrivilegeTemplate using conversion mechanism presumed to exist
#PrivilegeTemplateStrings[0] = $PrivilegeTemplateString;

$Role = smNewInstance (“CIM_Role”);
//build the parameter lists and invoke the method
%InArguments[] = {newArgument(“RoleTemplate”, $RoleTemplateString),
                  newArgument(“OwningSystem”, $system.GetObjectPath()),
                  newArgument(“Privileges”, #PrivilegeTemplateStrings[])};
%OutArguments[] = { newArgument(“Role”,
                     $Role.GetObjectPath()) };
//invoke method
#returnStatus = smOpInvokeMethod ($Service.GetObjectPath(),
   “CreateRole”,
   %InArguments[],
   %OutArguments[]);
//3 process return code to CLP Command Status
if (0 != #Error.code)  {
   //method invocation failed
   if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )    {
      // if the method invocation contains an embedded error
      // use it for the Error for the overall job
      &smAddError($job, #Error.$error[0]);
      &smMakeCommandStatus($job);
      &smEnd;
   }
```

```
   else if (#Error.code == 17)   {
   //trap for CIM_METHOD_NOT_FOUND
   //and make nice Unsupported msg.
      //unsupported
      $OperationError = smNewInstance("CIM_Error");
      //CIM_ERR_NOT_SUPPORTED
      $OperationError.CIMStatusCode = 7;
      //Other
      $OperationError.ErrorType = 1;
      //Low
      $OperationError.PerceivedSeverity = 2;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000001;
      $OperationError.Message = "Operation is not supported.";
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else {
      //operation failed, but no detailed error instance, need to make one up
      //make an Error instance and associate with job for Operation
         $OperationError = smNewInstance("CIM_Error");
         //CIM_ERR_FAILED
         $OperationError.CIMStatusCode = 1;
         //Software Error
         $OperationError.ErrorType = 4;
         //Unknown
         $OperationError.PerceivedSeverity = 0;
         $OperationError.OwningEntity = DMTF:SMCLP;
         $OperationError.MessageID = 0x00000009;
         $OperationError.Message = "An internal software error has occurred.";
         &smAddError($job, $OperationError);
         &smMakeCommandStatus($job);
         &smEnd;
   }
}//if CIM op failed
else if (0 == #returnStatus)  {
   //completed successfully, show created instance
   &lShowRole($Role, #all);
   &smEnd;
}
else if (4 == #returnStatus)  {
   //generic failure
   $OperationError = smNewInstance("CIM_Error");
   //CIM_ERR_FAILED
   $OperationError.CIMStatusCode = 1;
   //Other
   $OperationError.ErrorType = 1;
   //Low
```

```
   $OperationError.PerceivedSeverity = 2;
   $OperationError.OwningEntity = DMTF:SMCLP;
   $OperationError.MessageID = 0x00000002;
   $OperationError.Message = "Failed.  No further information is available.";
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
}
else   {
   //invalid parameter
   $OperationError = smNewInstance("CIM_Error");
   //CIM_ERR_FAILED
   $OperationError.CIMStatusCode = 1;
   //Other
   $OperationError.ErrorType = 1;
   //Low
   $OperationError.PerceivedSeverity = 2;
   $OperationError.OwningEntity = DMTF:SMCLP;
   $OperationError.MessageID = 0x00000004;
   $OperationError.Message = "One or more parameters specified are invalid.";
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
   &smEnd;
}
```

6.7.3 Delete

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

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

6.7.3.1 Delete a Single Instance

Delete a single instance of CIM_Role.

6.7.3.1.1 Command Form

```
delete <CIM_Role single instance>
```

6.7.3.1.2 CIM Requirements

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

6.7.3.1.3 Behavior Requirements

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

```
        NULL,
        NULL,
        $SvcInstancePaths[])
if (0 != #Error.code)
{
       &smProcessOpError (#Error);
       //includes &smEnd;
}
$service = $SvcInstancePaths[0];
//build the parameter lists and invoke the method
%InArguments[] = {newArgument(“Role”, $Role->)};
%OutArguments[] = { };
//invoke method
#returnStatus = smOpInvokeMethod ($Service.GetObjectPath(),
   “DeleteRole”,
   %InArguments[],
   %OutArguments[]);
//3 process return code to CLP Command Status
if (0 != #Error.code)  {
   //method invocation failed
   if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )    {
      // if the method invocation contains an embedded error
      // use it for the Error for the overall job
      &smAddError($job, #Error.$error[0]);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else if (#Error.code == 17)   {
   //trap for CIM_METHOD_NOT_FOUND
   //and make nice Unsupported msg.
      //unsupported
      $OperationError = smNewInstance(“CIM_Error”);
      //CIM_ERR_NOT_SUPPORTED
      $OperationError.CIMStatusCode = 7;
      //Other
      $OperationError.ErrorType = 1;
      //Low
      $OperationError.PerceivedSeverity = 2;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000001;
      $OperationError.Message = “Operation is not supported.”;
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else {
      //operation failed, but no detailed error instance, need to make one up
      //make an Error instance and associate with job for Operation
         $OperationError = smNewInstance(“CIM_Error”);
```

```
          //CIM_ERR_FAILED
          $OperationError.CIMStatusCode = 1;
          //Software Error
          $OperationError.ErrorType = 4;
          //Unknown
          $OperationError.PerceivedSeverity = 0;
          $OperationError.OwningEntity = DMTF:SMCLP;
          $OperationError.MessageID = 0x00000009;
          $OperationError.Message = "An internal software error has occurred.";
          &smAddError($job, $OperationError);
          &smMakeCommandStatus($job);
          &smEnd;
   }
}//if CIM op failed
else {
   //completed successfully, show created instance
   &smCommandCompleted($job);
   &smEnd;
}
```

6.7.4 Set

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

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

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*.

6.7.4.1.1 Command Form

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

6.7.4.1.2 CIM Requirements

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

6.7.4.1.3 Behavior Requirements

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

6.7.5.1.3 Behavior Requirements

6.7.5.1.3.1 Preconditions

`$instance` represents the targeted instance of CIM_Role.

```
$instance=<CIM_Role single instance>
```

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

6.7.5.1.3.2 Pseudo Code

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

6.7.5.2 Show Multiple Instances

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

6.7.5.2.1 Command Form

```
show <CIM_Role multiple instances>
```

6.7.5.2.2 CIM Requirements

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

6.7.5.2.3 Behavior Requirements

6.7.5.2.3.1 Preconditions

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

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

6.7.5.2.3.2 Pseudo Code

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

```
   if (0 != #Error.code)
   {
   &smProcessOpError (#Error);
   //includes &smEnd;
   }
   for #i in $outInstancePaths->[]
   {
   &lShowRole($outInstancePaths->[i]);
   }
&smEnd;
```

6.8 CIM_RoleBasedManagementCapabilities

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

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

Table 9 – Command Verb Requirements for CIM_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	

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

6.8.1 Ordering of Results

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

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

6.8.2 Show

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

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

6.8.2.1 Show a Single Instance

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

6.8.2.1.1 Command Form

```
show <CIM_RoleBasedManagementCapabilities single instance>
```

6.8.2.1.2 CIM Requirements

See CIM_RoleBasedManagementCapabilities in the “CIM Elements” section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.8.2.1.3 Behavior Requirements

6.8.2.1.3.1 Preconditions

`$instance` represents the targeted instance of CIM_RoleBasedManagementCapabilities.

```
$instance=<CIM_RoleBasedManagementCapabilities single instance>
```

`#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

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

6.8.2.2.1 Command Form

```
show <CIM_RoleBasedManagementCapabilities multiple instances>
```

6.8.2.2.2 CIM Requirements

See CIM_RoleBasedManagementCapabilities in the “CIM Elements” section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.8.2.2.3 Behavior Requirements

6.8.2.2.3.1 Preconditions

`$containerInstance` represents the instance of CIM_ConcreteCollection with ElementName property that contains “Capabilities” and is associated with the targeted instances of CIM_RoleBasedManagementCapabilities through the CIM_MemberOfCollection association.

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

6.8.2.2.3.2 Pseudo Code

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

6.9 CIM_RoleBasedAuthorizationService

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

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

Table 10 – Command Verb Requirements for CIM_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	

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

6.9.1 Ordering of Results

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

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

6.9.2 Show

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

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

6.9.2.1 Show a Single Instance

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

6.9.2.1.1 Command Form

```
show <CIM_RoleBasedAuthorizationService single instance>
```

6.9.2.1.2 CIM Requirements

See CIM_RoleBasedAuthorizationService in the “CIM Elements” section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.9.2.1.3 Behavior Requirements

6.9.2.1.3.1 Preconditions

$instance represents the targeted instance of CIM_RoleBasedAuthorizationService.

```
$instance=<CIM_RoleBasedAuthorizationService single instance>
```

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

6.9.2.1.3.2 Pseudo Code

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

6.9.2.2 Show Multiple Instances

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

6.9.2.2.1 Command Form

```
show <CIM_RoleBasedAuthorizationService multiple instances>
```

6.9.2.2.2 CIM Requirements

See CIM_RoleBasedAuthorizationService in the “CIM Elements” section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.9.2.2.3 Behavior Requirements

6.9.2.2.3.1 Preconditions

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

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

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

6.9.2.2.3.2 Pseudo Code

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

6.10 CIM_RoleLimitedToTarget

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

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

Table 11 – Command Verb Requirements for CIM_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	

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

6.10.1 Ordering of Results

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

- 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;
```

```
   //Unknown
   $OperationError.PerceivedSeverity = 0;
   $OperationError.OwningEntity = DMTF:SMCLP;
   $OperationError.MessageID = 0x00000010;
   $OperationError.Message = “The target association can not be created between the
      specified targets.”;
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
   &smEnd;
}
#Error = &smOpAssociatorNames(
         $Role->,
         “CIM_RoleLimitedToTarget”,
         “CIM_ManagedElement”,
         NULL,
         NULL,
         NULL,
         $MEInstancePaths->[])
if (0 != #Error.code)
{
      &smProcessOpError (#Error);
      //includes &smEnd;
}
#MEInstancePaths->[#MEInstancePaths.length] = $ME->
#Error = &smOpAssociators(
         $Role->,
         “CIM_ServiceAffectsElement”,
         “CIM_RoleBasedAuthorizationService”,
         NULL,
         NULL,
         NULL,
         $SvcInstancePaths[])
if (0 != #Error.code)
{
      &smProcessOpError (#Error);
      //includes &smEnd;
}
// invoke the method
%InArguments[] = { newArgument(“Role”, $Role->),
newArgument(“RoleLimitedToTargets”, #MEInstancePaths->[]) }
%OutArguments[] = {  };
//invoke method
#returnStatus = smOpInvokeMethod ($SvcInstancePaths[0].GetObjectPath(),
   “ModifyRole”,
   %InArguments[],
   %OutArguments[]);
//3 process return code to CLP Command Status
if (0 != #Error.code)  {
   //method invocation failed
```

```
   if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )    {
      // if the method invocation contains an embedded error
      // use it for the Error for the overall job
      &smAddError($job, #Error.$error[0]);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else if (#Error.code == 17)   {
   //trap for CIM_METHOD_NOT_FOUND
   //and make nice Unsupported msg.
      //unsupported
      $OperationError = smNewInstance("CIM_Error");
      //CIM_ERR_NOT_SUPPORTED
      $OperationError.CIMStatusCode = 7;
      //Other
      $OperationError.ErrorType = 1;
      //Low
      $OperationError.PerceivedSeverity = 2;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000001;
      $OperationError.Message = "Operation is not supported.";
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else if (#Error.code != 0) {
      //operation failed, but no detailed error instance, need to make one up
      //make an Error instance and associate with job for Operation
         $OperationError = smNewInstance("CIM_Error");
         //CIM_ERR_FAILED
         $OperationError.CIMStatusCode = 1;
         //Software Error
         $OperationError.ErrorType = 4;
         //Unknown
         $OperationError.PerceivedSeverity = 0;
         $OperationError.OwningEntity = DMTF:SMCLP;
         $OperationError.MessageID = 0x00000009;
         $OperationError.Message = "An internal software error has occurred.";
         &smAddError($job, $OperationError);
         &smMakeCommandStatus($job);
         &smEnd;
   }
}//if CIM op failed
//modify worked, show the resultant association instance
   &smShowAssociationInstance ( "CIM_RoleLimitedToTarget", $instance1->,
      $instance2->);
   &smEnd;
}
```

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;
```

```
   $OperationError.OwningEntity = DMTF:SMCLP;
   $OperationError.MessageID = 0x00000010;
   $OperationError.Message = “The target association can not be Deleted between the
      specified targets.”;
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
   &smEnd;
}
#Error = &smOpAssociatorNames(
        $Role->,
        “CIM_RoleLimitedToTarget”,
        “CIM_ManagedElement”,
        NULL,
        NULL,
        NULL,
        $MEInstancePaths->[])
if (0 != #Error.code)
{
   &smProcessOpError (#Error);
   //includes &smEnd;
}
for (#i=0; #i<$MEInstancePaths->[].length; #i++) {
if (#MEInstancePaths->[#i] == $ME->)  {
   #MEInstancePaths->[#i] = NULL;
}
#Error = &smOpAssociators(
         $Role->,
         “CIM_ServiceAffectsElement”,
         “CIM_RoleBasedAuthorizationService”,
         NULL,
         NULL,
         NULL,
         $SvcInstancePaths[])
if (0 != #Error.code)
{
   &smProcessOpError (#Error);
   //includes &smEnd;
}
// invoke the method
%InArguments[] = { newArgument(“Role”, $Role->),
newArgument(“RoleLimitedToTargets”, #MEInstancePaths->[]) }
%OutArguments[] = {  };
//invoke method
#returnStatus = smOpInvokeMethod ($SvcInstancePaths[0].GetObjectPath(),
   “ModifyRole”,
   %InArguments[],
   %OutArguments[]);
//3 process return code to CLP Command Status
if (0 != #Error.code)  {
```

```
   //method invocation failed
   if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) ) {
      // if the method invocation contains an embedded error
      // use it for the Error for the overall job
      &smAddError($job, #Error.$error[0]);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else if (#Error.code == 17) {
   //trap for CIM_METHOD_NOT_FOUND
   //and make nice Unsupported msg.
      //unsupported
      $OperationError = smNewInstance("CIM_Error");
      //CIM_ERR_NOT_SUPPORTED
      $OperationError.CIMStatusCode = 7;
      //Other
      $OperationError.ErrorType = 1;
      //Low
      $OperationError.PerceivedSeverity = 2;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000001;
      $OperationError.Message = "Operation is not supported.";
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else if (#Error.code != 0) {
      //operation failed, but no detailed error instance, need to make one up
      //make an Error instance and associate with job for Operation
         $OperationError = smNewInstance("CIM_Error");
         //CIM_ERR_FAILED
         $OperationError.CIMStatusCode = 1;
         //Software Error
         $OperationError.ErrorType = 4;
         //Unknown
         $OperationError.PerceivedSeverity = 0;
         $OperationError.OwningEntity = DMTF:SMCLP;
         $OperationError.MessageID = 0x00000009;
         $OperationError.Message = "An internal software error has occurred.";
         &smAddError($job, $OperationError);
         &smMakeCommandStatus($job);
         &smEnd;
      }
}//if CIM op failed
//modify worked, show the resultant association instance
   &smShowAssociationInstance ( "CIM_RoleLimitedToTarget", $instance1->,
      $instance2->);
   &smEnd;
}
```

6.10.4 Show

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

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

6.10.4.1 Show Multiple Instances – CIM_Role Reference

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

6.10.4.1.1 Command Form

```
show <CIM_RoleLimitedToTarget multiple instances>
```

6.10.4.1.2 CIM Requirements

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

6.10.4.1.3 Behavior Requirements

6.10.4.1.3.1 Preconditions

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

6.10.4.1.3.2 Pseudo Code

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

6.10.4.2 Show Multiple Instances – CIM_ManagedElement reference

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

6.10.4.2.1 Command Form

```
show <CIM_RoleLimitedToTarget multiple instances>
```

6.10.4.2.2 CIM Requirements

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

6.10.4.2.3 Behavior Requirements

6.10.4.2.3.1 Preconditions

`$instance` contains the instance of CIM_ManagedElement subclass that is referenced by 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	

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

6.11.1 Ordering of Results

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

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

6.11.2 Show

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

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

6.11.2.1 Show Multiple Instances

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

6.11.2.1.1 Command Form

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

6.11.2.1.2 CIM Requirements

See CIM_ServiceAffectsElement in the “CIM Elements” section of the *Role Based Authorization Profile* for the list of mandatory properties.

6.11.2.1.3 Behavior Requirements

6.11.2.1.3.1 Preconditions

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

6.11.2.1.3.2 Pseudo Code

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

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

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

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	

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

6.12.1 Ordering of Results

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

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

6.12.2 Show

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

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

6.12.2.1 Show Multiple Instances – CIM_RoleBasedAuthorizationService

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_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;
```

6.12.2.3 Show a Single Instances – Both References

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 a reference is specified to an instance of CIM_AccountManagementService and the other reference is to an instance of CIM_RoleBasedAuthorizationService.

6.12.2.3.1 Command Form

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

6.12.2.3.2 CIM Requirements

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

6.12.2.3.3 Behavior Requirements

6.12.2.3.3.1 Preconditions

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

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

6.12.2.3.3.2 Pseudo Code

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

ANNEX A
(informative)

Change Log

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