Document Number: DSP0826

Date: 2009-07-14

Version: 1.0.0

Software Inventory Profile SM CLP Mapping Specification

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

CONTENTS

Tables

67

Foreword

The *Software Inventory Profile SM CLP Mapping Specification* (DSP0826) was prepared by the Server Management Working Group.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability.

Conventions

The pseudo-code conventions utilized in this document are the Recipe Conventions as defined in SNIA SMI-S V1.1.0, section 7.6.

Acknowledgements

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

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

Introduction

This document defines the SM CLP mapping for CIM elements described in the *Software Inventory Profile*. The information in this specification, combined with *SM CLP-to-CIM Common Mapping Specification V1.0* (DSP0216), is intended to be sufficient to implement SM CLP commands relevant to the classes, properties, and methods described in the *Software Inventory Profile* using CIM operations.

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

Software Inventory Profile SM CLP Mapping Specification

1 Scope

This specification contains the requirements for an implementation of the SM CLP to provide access to, and implement the behaviors of, the *Software Inventory 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 DSP1023, *Software Inventory Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1023_1.0.pdf

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

2.2 Other References

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

3 Terms and Definitions

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

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

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

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

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

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

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

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

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

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

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

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

4 Symbols and Abbreviated Terms

The following symbols and abbreviations are used in this document.

4.1
CIM
Common Information Model

4.2
CLP
Command Line Protocol

4.3
DMTF
Distributed Management Task Force

4.4
IETF
Internet Engineering Task Force

4.5
SM
Server Management

4.6
SMI-S
Storage Management Initiative Specification

4.7
SNIA
Storage Networking Industry Association

4.8
UFsT
User Friendly selection Tag

5 Recipes

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

- smShowInstance
- smShowInstances
- smShowAssociationInstance
- smShowAssociationInstances

There are no Local Recipes defined for use in this mapping.

6 Mappings

The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in the *Software Inventory 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_SoftwareIdentity

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

Table 1 lists each SM CLP verb, the required level of support for the verb in conjunction with the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and target. Table 1 is for informational purposes only; in case of a conflict between Table 1 and requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 1.

Table 1 – Command Verb Requirements for CIM_SoftwareIdentity

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

Command Verb	Requirement	Comments
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`, `exit`, `load`, `reset`, `set`, `start`, and `stop`.

6.1.1 Ordering of Results

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

- Results for CIM_SoftwareIdentity 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_SoftwareIdentity. Implementations shall support the use of the `show` verb with CIM_SoftwareIdentity.

6.1.2.1 Show Command Form for a Single Object Target – CIM_SoftwareIdentity

This command form is used to show a single instance of CIM_SoftwareIdentity. This command form corresponds to a `show` command issued against a single instance of CIM_ SoftwareIdentity where only one reference is specified and the reference is to an instance of CIM_SoftwareIdentity.

6.1.2.1.1 Command Form

```
show <CIM_SoftwareIdentity single object>
```

6.1.2.1.2 CIM Requirements

See CIM_SoftwareIdentity in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.1.2.1.3 Behavior Requirements

6.1.2.1.3.1 Preconditions

`$instance` represents the targeted instance of CIM_SoftwareIdentity.

```
$instance=<CIM_SoftwareIdentity single object>;
```

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

6.1.2.1.3.2 Pseudo Code

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

6.1.2.2 Show Command Form for Multiple Objects Target for the Installed Software

6.1.2.2.1 Command Form

```
show <CIM_SoftwareIdentity multiple objects>
```

6.1.2.2.2 CIM Requirements

See CIM_SoftwareIdentity in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.1.2.2.3 Behavior Requirements

6.1.2.2.3.1 Preconditions

`$containerInstance` is the instance of CIM_ComputerSystem or CIM_System that is associated with the targeted instances of CIM_SoftwareIdentity through the CIM_InstalledSoftwareIdentity association.

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

6.1.2.2.3.2 Pseudo Code

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

6.1.2.3 Show Command Form for Multiple Objects Target for the Available Software

6.1.2.3.1 Command Form

```
show <CIM_SoftwareIdentity multiple objects>
```

6.1.2.3.2 CIM Requirements

See CIM_SoftwareIdentity in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.1.2.3.3 Behavior Requirements

6.1.2.3.3.1 Preconditions

`$containerInstance` contains the instance of CIM_SystemSpecificCollection that is associated with the targeted instances of CIM_SoftwareIdentity through the CIM_MemberOfCollection association.

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

6.1.2.3.3.2 Pseudo Code

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

6.1.2.4 Show Command Form for Multiple Objects Target for a Software Bundle

6.1.2.4.1 Command Form

```
show <CIM_SoftwareIdentity multiple objects>
```

6.1.2.4.2 CIM Requirements

See CIM_SoftwareIdentity in the "CIM Elements" section of the *Software Inventory Profile* for the list of mandatory properties.

6.1.2.4.3 Behavior Requirements

6.1.2.4.3.1 Preconditions

`$containerInstance` contains the instance of CIM_SoftwareIdentity that is associated with the targeted instances of CIM_SoftwareIdentity through the CIM_OrderedComponent association.

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

6.1.2.4.3.2 Pseudo Code

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

6.1.2.5 Show Command Form for Multiple Objects Target for a Software Dependency

6.1.2.5.1 Command Form

```
show <CIM_SoftwareIdentity multiple objects>
```

6.1.2.5.2 CIM Requirements

See CIM_SoftwareIdentity in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.1.2.5.3 Behavior Requirements

6.1.2.5.3.1 Preconditions

`$containerInstance` contains the instance of CIM_SoftwareIdentity that is associated with the targeted instances of CIM_SoftwareIdentity through the CIM_OrderedDependency association.

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

6.1.2.5.3.2 Pseudo Code

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

6.2 CIM_InstalledSoftwareIdentity

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

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

Table 2 – Command Verb Requirements for CIM_InstalledSoftwareIdentity

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 6.2.2.
start	Not supported	
stop	Not supported	

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

6.2.1 Ordering of Results

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

- Results for CIM_InstalledSoftwareIdentity 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_InstalledSoftwareIdentity. Implementations shall support the use of the `show` verb with CIM_InstalledSoftwareIdentity.

6.2.2.1 Show Command Form for Multiple Objects Target – CIM_System or CIM_ComputerSystem Reference

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

6.2.2.1.1 Command Form

```
show <CIM_InstalledSoftwareIdentity multiple objects>
```

6.2.2.1.2 CIM Requirements

See CIM_InstalledSoftwareIdentity in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.2.2.1.3 Behavior Requirements

6.2.2.1.3.1 Preconditions

`$instance` represents the instance of a CIM_System or CIM_ComputerSystem, which is referenced by CIM_InstalledSoftwareIdentity.

```
$instance=<CIM_ComputerSystem single object>;
```

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

6.2.2.1.3.2 Pseudo Code

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

6.2.2.2 Show Command Form for Multiple Objects Target – CIM_SoftwareIdentity Reference

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

6.2.2.2.1 Command Form

```
show <CIM_InstalledSoftwareIdentity multiple objects>
```

6.2.2.2.2 CIM Requirements

See CIM_InstalledSoftwareIdentity in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.2.2.2.3 Behavior Requirements

6.2.2.2.3.1 Preconditions

`$instance` represents the instance of a CIM_SoftwareIdentity, which is referenced by CIM_InstalledSoftwareIdentity.

```
$instance=<CIM_SoftwareIdentity single object>
```

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

6.2.2.2.3.2 Pseudo Code

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

6.2.2.3 Show Command Form for a Single Object Target – 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_InstalledSoftwareIdentity where both references are specified and therefore the desired instance is unambiguously identified.

6.2.2.3.1 Command Form

```
show <CIM_InstalledSoftwareIdentity single object>
```

6.2.2.3.2 CIM Requirements

See CIM_InstalledSoftwareIdentity in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.2.2.3.3 Behavior Requirements

6.2.2.3.3.1 Preconditions

`$instanceA` represents the instance of CIM_System or CIM_ComputerSystem and `$instanceB` represents the instance of CIM_SoftwareIdentity, both of which are referenced by CIM_InstalledSoftwareIdentity.

```
$instanceA=<CIM_ComputerSystem single object>;
$instanceB=<CIM_SoftwareIdentity single object>;
```

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

6.2.2.3.3.2 Pseudo Code

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

6.3 CIM_ElementSoftwareIdentity

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

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

Table 3 – Command Verb Requirements for CIM_ElementSoftwareIdentity

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

Command Verb	Requirement	Comments
reset	Not supported	
set	Not supported	
show	Shall	See 6.3.2.
start	Not supported	
stop	Not supported	

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

6.3.1 Ordering of Results

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

- Results for CIM_ElementSoftwareIdentity 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_ElementSoftwareIdentity. Implementations shall support the use of the `show` verb with CIM_ElementSoftwareIdentity.

6.3.2.1 Show Command Form for Multiple Objects Target – CIM_ManagedElement Reference

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

6.3.2.1.1 Command Form

```
show <CIM_ElementSoftwareIdentity multiple objects>
```

6.3.2.1.2 CIM Requirements

See CIM_ElementSoftwareIdentity in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.3.2.1.3 Behavior Requirements

6.3.2.1.3.1 Preconditions

`$instance` represents the instance of a CIM_ManagedElement, which is referenced by CIM_ElementSoftwareIdentity.

```
$instance=<CIM_ManagedElement single object>;
```

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

6.3.2.1.3.2 Pseudo Code

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

6.3.2.2 Show Command Form for Multiple Objects – CIM_SoftwareIdentity Reference

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

6.3.2.2.1 Command Form

```
show <CIM_ElementSoftwareIdentity multiple objects>
```

6.3.2.2.2 CIM Requirements

See CIM_ElementSoftwareIdentity in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.3.2.2.3 Behavior Requirements

6.3.2.2.3.1 Preconditions

`$instance` represents the instance of a CIM_SoftwareIdentity, which is referenced by CIM_ElementSoftwareIdentity.

```
$instance=<CIM_SoftwareIdentity single object>
```

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

6.3.2.2.3.2 Pseudo Code

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

6.3.2.3 Show Command Form for a Single Object Target – 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_ElementSoftwareIdentity where both references are specified and therefore the desired instance is unambiguously identified.

6.3.2.3.1 Command Form

```
show <CIM_ElementSoftwareIdentity single object>
```

6.3.2.3.2 CIM Requirements

See CIM_ElementSoftwareIdentity in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.3.2.3.3 Behavior Requirements

6.3.2.3.3.1 Preconditions

`$instanceA` represents the instance of a CIM_ManagedElement and `$instanceB` represents the instance of CIM_SoftwareIdentity, both of which are referenced by CIM_ElementSoftwareIdentity.

```
$instanceA=<CIM_ManagedElement single object>;
$instanceB=<CIM_SoftwareIdentity single object>;
```

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

6.3.2.3.3.2 Pseudo Code

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

6.4 CIM_SystemSpecificCollection

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

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

Table 4 – Command Verb Requirements for CIM_SystemSpecificCollection

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 6.4.2.
start	Not supported	
stop	Not supported	

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

6.4.1 Ordering of Results

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

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

6.4.2 Show

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

6.4.2.1 Show Command Form for a Single Object Target

6.4.2.1.1 Command Form

```
show <CIM_SystemSpecificCollection single object>
```

6.4.2.1.2 CIM Requirements

See CIM_SystemSpecificCollection in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.4.2.1.3 Behavior Requirements

6.4.2.1.3.1 Preconditions

`$instance` represents the targeted instance of CIM_SystemSpecificCollection.

```
$instance=<CIM_SystemSpecificCollection single object>;
```

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

6.4.2.1.3.2 Pseudo Code

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

6.4.2.2 Show Command Form for a Single Object Target – CIM_ComputerSystem or CIM_System Reference

6.4.2.2.1 Command Form

```
show <CIM_SystemSpecificCollection single object>
```

6.4.2.2.2 CIM Requirements

See CIM_SystemSpecificCollection in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.4.2.2.3 Behavior Requirements

6.4.2.2.3.1 Preconditions

`$containerInstance` contains the instance of CIM_ComputerSystem that is associated with the targeted instances of CIM_SystemSpecificCollection through the CIM_HostedCollection association.

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

6.4.2.2.3.2 Pseudo Code

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

6.4.2.3 Show Command Form for Multiple Objects Target - CIM_SoftwareIdentity Reference

6.4.2.3.1 Command Form

```
show <CIM_SystemSpecificCollection multiple objects>
```

6.4.2.3.2 CIM Requirements

See CIM_SystemSpecificCollection in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.4.2.3.3 Behavior Requirements

6.4.2.3.3.1 Preconditions

`$containerInstance` contains the instance of CIM_SoftwareIdentity that is associated with the targeted instances of CIM_SystemSpecificCollection through the CIM_MemberOfCollection association.

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

6.4.2.3.3.2 Pseudo Code

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

6.5 CIM_HostedCollection

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

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

Table 5 – Command Verb Requirements for CIM_HostedCollection

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.5.2.
start	Not supported	
stop	Not supported	

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

6.5.1 Ordering of Results

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

- Results for CIM_HostedCollection 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_HostedCollection. Implementations shall support the use of the `show` verb with CIM_HostedCollection.

6.5.2.1 Show Command Form for a Single Object Target – CIM_ComputerSystem or CIM_System Reference

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

6.5.2.1.1 Command Form

```
show <CIM_HostedCollection single object>
```

6.5.2.1.2 CIM Requirements

See CIM_HostedCollection in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.5.2.1.3 Behavior Requirements

6.5.2.1.3.1 Preconditions

`$instance` represents the instance of CIM_ComputerSystem or CIM_System, which is referenced by CIM_HostedCollection.

```
$instance=<CIM_ComputerSystem single object>;
```

`#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[] = <array of mandatory non-key property names (see CIM
      Requirements)>;
   }
&smShowAssociationInstances ( “CIM_HostedCollection”, $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.5.2.2 Show Command Form for a Single Object – CIM_SystemSpecificCollection Reference

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

6.5.2.2.1 Command Form

```
show <CIM_HostedCollection single object>
```

6.5.2.2.2 CIM Requirements

See CIM_HostedCollection in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.5.2.2.3 Behavior Requirements

6.5.2.2.3.1 Preconditions

`$instance` represents the instance of a CIM_SystemSpecificCollection, which is referenced by CIM_HostedCollection.

```
$instance=<CIM_SystemSpecificCollection single object>
```

#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[] = <array of mandatory non-key property names (see CIM
      Requirements)>;
   }
&smShowAssociationInstances ( “CIM_HostedCollection”, $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.5.2.3 Show Command Form for a Single Object Target – 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_HostedCollection where both references are specified and therefore the desired instance is unambiguously identified.

6.5.2.3.1 Command Form

```
show <CIM_HostedCollection single object>
```

6.5.2.3.2 CIM Requirements

See CIM_HostedCollection in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.5.2.3.3 Behavior Requirements

6.5.2.3.3.1 Preconditions

$instanceA represents the instance of CIM_ComputerSystem or CIM_System and $instanceB represents the instance of CIM_SystemSpecificCollection, both of which are referenced by CIM_HostedCollection.

```
$instanceA=<CIM_ComputerSystem single object>;
$instanceB=<CIM_SystemSpecificCollection single object>;
```

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

6.5.2.3.3.2 Pseudo Code

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

6.6 CIM_MemberOfCollection

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

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

Table 6 – Command Verb Requirements for CIM_MemberOfCollection

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

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

6.6.1 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.

6.6.2 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.

6.6.2.1 Show Command Form for Multiple Objects Target – CIM_SystemSpecificCollection Reference

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

6.6.2.1.1 Command Form

```
show <CIM_MemberOfCollection multiple objects>
```

6.6.2.1.2 CIM Requirements

See CIM_MemberOfCollection in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.6.2.1.3 Behavior Requirements

6.6.2.1.3.1 Preconditions

`$instance` represents the instance of a CIM_SystemSpecificCollection, which is referenced by CIM_MemberOfCollection.

```
$instance=<CIM_SystemSpecificCollection single object>;
```

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

6.6.2.1.3.2 Pseudo Code

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

6.6.2.2 Show Command Form for Multiple Objects Target – CIM_SoftwareIdentity Reference

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

6.6.2.2.1 Command Form

```
show <CIM_MemberOfCollection multiple objects>
```

6.6.2.2.2 CIM Requirements

See CIM_MemberOfCollection in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.6.2.2.3 Behavior Requirements

6.6.2.2.3.1 Preconditions

`$instance` represents the instance of a CIM_SoftwareIdentity, which is referenced by CIM_MemberOfCollection.

```
$instance = <CIM_SoftwareIdentity single object>
```

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

6.6.2.2.3.2 Pseudo Code

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

6.6.2.3 Show Command Form for a Single Object Target – 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_MemberOfCollection where both references are specified and therefore the desired instance is unambiguously identified.

6.6.2.3.1 Command Form

```
show <CIM_MemberOfCollection single object>
```

6.6.2.3.2 CIM Requirements

See CIM_MemberOfCollection in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.6.2.3.3 Behavior Requirements

6.6.2.3.3.1 Preconditions

`$instanceA` represents the instance of a CIM_SystemSpecificCollection and `$instanceB` represents the instance of CIM_SoftwareIdentity, both of which are referenced by CIM_MemberOfCollection.

```
$instanceA=<CIM_SystemSpecificCollection single object>;
$instanceB=<CIM_SoftwareIdentity single object>;
```

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

6.6.2.3.3.2 Pseudo Code

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

6.7 CIM_SoftwareIdentityResource

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

Table 7 lists each SM CLP verb, the required level of support for the verb in conjunction with the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and

target. Table 7 is for informational purposes only; in case of a conflict between Table 7 and requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 7.

Table 7 – Command Verb Requirements for CIM_SoftwareIdentityResource

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

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

6.7.1 Ordering of Results

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

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

6.7.2 Show

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

6.7.2.1 Show Command Form for a Single Object Target

6.7.2.1.1 Command Form

```
show <CIM_SoftwareIdentityResource single object>
```

6.7.2.1.2 CIM Requirements

See CIM_SoftwareIdentityResource in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.7.2.1.3 Behavior Requirements

6.7.2.1.3.1 Preconditions

`$instance` represents the targeted instance of CIM_SoftwareIdentityResource.

```
$instance=<CIM_SoftwareIdentityResource single object>;
```

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

6.7.2.1.3.2 Pseudo Code

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

6.7.2.2 Show Command Form for Multiple Objects Target

6.7.2.2.1 Command Form

```
show <CIM_SoftwareIdentityResource multiple objects>
```

6.7.2.2.2 CIM Requirements

See CIM_SoftwareIdentityResource in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.7.2.2.3 Behavior Requirements

6.7.2.2.3.1 Preconditions

`$containerInstance` contains the instance of CIM_ComputerSystem that is associated with the targeted instances of CIM_SoftwareIdentityResource through the CIM_HostedAccessPoint association.

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

6.7.2.2.3.2 Pseudo Code

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

6.8 CIM_SAPAvailableForElement

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

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

843 **Table 8 – Command Verb Requirements for CIM_SAPAvailableForElement**

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	

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

846 ### 6.8.1 Ordering of Results

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

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

850 ### 6.8.2 Show

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

854 #### 6.8.2.1 Show Command Form for a Single Object Target – CIM_SoftwareIdentity Reference

855 This command form is used to show a single instance of CIM_SAPAvailableForElement. This command
856 form corresponds to a `show` command issued against a single instance of CIM_SAPAvailableForElement
857 where only one reference is specified and the reference is to the scoping instance of
858 CIM_SoftwareIdentity.

859 ##### 6.8.2.1.1 Command Form

860 ```
show <CIM_SAPAvailableForElement single object>
```

861 ##### 6.8.2.1.2 CIM Requirements

862 See CIM_SAPAvailableForElement in the “CIM Elements” section of the *Software Inventory Profile* for the
863 list of mandatory properties.

864 ##### 6.8.2.1.3 Behavior Requirements

865 ###### 6.8.2.1.3.1 Preconditions

866 `$instance` represents the instance of a CIM_SoftwareIdentity, which is referenced by
867 CIM_SAPAvailableForElement.
```

```
$instance=<CIM_SoftwareIdentity single object>
```

#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[] = <array of mandatory non-key property names (see CIM
      Requirements)>;
   }
&smShowAssociationInstances ( “CIM_SAPAvailableForElement”, $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.8.2.2 Show Command Form for a Single Object Target – CIM_SoftwareIdentityResource Reference

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

6.8.2.2.1 Command Form

```
show <CIM_SAPAvailableForElement single object>
```

6.8.2.2.2 CIM Requirements

See CIM_SAPAvailableForElement in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.8.2.2.3 Behavior Requirements

6.8.2.2.3.1 Preconditions

$instance represents the instance of a CIM_SoftwareIdentityResource, which is referenced by CIM_SAPAvailableForElement.

```
$instance=<CIM_SoftwareIdentityResource single object>
```

#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[] = <array of mandatory non-key property names (see CIM
      Requirements)>;
   }
&smShowAssociationInstances ( “CIM_SAPAvailableForElement”, $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.8.2.3 Show Command Form for a Single Object Target – 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_SAPAvailableForElement where both references are specified and therefore the desired instance is unambiguously identified.

6.8.2.3.1 Command Form

```
show < CIM_SAPAvailableForElement single object>
```

6.8.2.3.2 CIM Requirements

See CIM_SAPAvailableForElement in the "CIM Elements" section of the *Software Inventory Profile* for the list of mandatory properties.

6.8.2.3.3 Behavior Requirements

6.8.2.3.3.1 Preconditions

`$instanceA` represents the instance of a CIM_SoftwareIdentity and `$instanceB` represents the instance of CIM_SoftwareIdentityResource, both of which are referenced by CIM_SAPAvailableForElement.

```
$instanceA=<CIM_SoftwareIdentity single object>;
$instanceB=<CIM_SoftwareIdentityResource single object>;
```

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

6.8.2.3.3.2 Pseudo Code

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

6.9 CIM_HostedAccessPoint

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

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

Table 9 – Command Verb Requirements for CIM_HostedAccessPoint

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

Command Verb	Requirement	Comments
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`, `start`, and `stop`.

6.9.1 Ordering of Results

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

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

6.9.2 Show

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

6.9.2.1 Show Command Form for Multiple Objects Target – CIM_ComputerSystem or CIM_System Reference

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

6.9.2.1.1 Command Form

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

6.9.2.1.2 CIM Requirements

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

6.9.2.1.3 Behavior Requirements

6.9.2.1.3.1 Preconditions

`$instance` represents the instance of CIM_ComputerSystem or CIM_System, which is referenced by CIM_HostedAccessPoint.

```
$instance=<CIM_ComputerSystem single object>;
```

`#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[] = <array of mandatory non-key property names (see CIM
      Requirements)>;
   }
&smShowAssociationInstances ( “CIM_HostedAccessPoint”, $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.9.2.2 Show Command Form for a Single Object Target – CIM_SoftwareIdentityResource Reference

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

6.9.2.2.1 Command Form

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

6.9.2.2.2 CIM Requirements

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

6.9.2.2.3 Behavior Requirements

6.9.2.2.3.1 Preconditions

`$instance` represents the instance of CIM_SoftwareIdentityResource, which is referenced by CIM_HostedAccessPoint.

```
$instance=<CIM_SoftwareIdentityResource single object>;
```

`#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[] = <array of mandatory non-key property names (see CIM
      Requirements)>;
   }
&smShowAssociationInstances ( “CIM_HostedAccessPoint”, $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.9.2.3 Show Command Form for a Single Object Target – 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_HostedAccessPoint where both references are specified and therefore the desired instance is unambiguously identified.

6.9.2.3.1 Command Form

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

6.9.2.3.2 CIM Requirements

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

6.9.2.3.3 Behavior Requirements

`$instanceA` represents the instance of a CIM_System or CIM_ComputerSystem and `$instanceB` represents the instance of CIM_SoftwareIdentityResource, both of which are referenced by CIM_HostedAccessPoint.

```
$instanceA=<CIM_ComputerSystem single object>;
$instanceB=<CIM_SoftwareIdentityResource single object>;
```

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

6.9.2.3.3.1 Pseudo Code

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

6.10 CIM_OrderedComponent

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

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

Table 10 – Command Verb Requirements for CIM_OrderedComponent

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

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

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

6.10.1 Ordering of Results

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

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

6.10.2 Show

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

6.10.2.1 Show Command Form for Multiple Objects Target – CIM_SoftwareIdentity Reference

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

6.10.2.1.1 Command Form

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

6.10.2.1.2 CIM Requirements

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

6.10.2.1.3 Behavior Requirements

6.10.2.1.3.1 Preconditions

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

```
$instance=<CIM_SoftwareIdentity single object>
```

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

6.10.2.1.3.2 Pseudo Code

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

6.10.2.2 Show Command Form for a Single Object Target – 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_OrderedComponent where both references are specified and therefore the desired instance is unambiguously identified.

6.10.2.2.1 Command Form

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

6.10.2.2.2 CIM Requirements

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

6.10.2.2.3 Behavior Requirements

`$instanceA` represents the instance of a CIM_SoftwareIdentity and `$instanceB` represents the instance of CIM_SoftwareIdentity, both of which are referenced by CIM_OrderedComponent.

```
$instanceA=<CIM_SoftwareIdentity single object>;
$instanceB=<CIM_SoftwareIdentity single object>;
```

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

6.10.2.2.3.1 Pseudo Code

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

6.11 CIM_OrderedDependency

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

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

requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 11.

Table 11 – Command Verb Requirements for CIM_OrderedDependency

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.11.2.
start	Not supported	
stop	Not supported	

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

6.11.1 Ordering of Results

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

- Results for CIM_OrderedDependency 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_OrderedDependency. Implementations shall support the use of the `show` verb with CIM_OrderedDependency.

6.11.2.1 Show Command Form for Multiple Objects Target – CIM_SoftwareIdentity Reference

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

6.11.2.1.1 Command Form

```
show <CIM_OrderedDependency multiple objects>
```

6.11.2.1.2 CIM Requirements

See CIM_OrderedDependency in the “CIM Elements” section of the *Software Inventory Profile* for the list of mandatory properties.

6.11.2.1.3 Behavior Requirements

6.11.2.1.3.1 Preconditions

$instance represents the instance of a CIM_SoftwareIdentity, which is referenced by CIM_OrderedDependency.

```
$instance=<CIM_SoftwareIdentity single object>
```

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

6.11.2.1.3.2 Pseudo Code

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

6.11.2.2 Show Command Form for a Single Object Target – 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_OrderedDependency where both references are specified and therefore the desired instance is unambiguously identified.

6.11.2.2.1 Command Form

```
show <CIM_OrderedDependency single object>
```

6.11.2.2.2 CIM Requirements

See CIM_OrderedDependency in the "CIM Elements" section of the *Software Inventory Profile* for the list of mandatory properties.

6.11.2.2.3 Behavior Requirements

$instanceA represents the instance of a CIM_SoftwareIdentity and $instanceB represents the instance of CIM_SoftwareIdentity, both of which are referenced by CIM_OrderedDependency.

```
$instanceA=<CIM_SoftwareIdentity single object>;
$instanceB=<CIM_SoftwareIdentity single object>;
```

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

6.11.2.2.3.1 Pseudo Code

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

ANNEX A
(informative)

Change Log

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