Document Number: DSP0810

Date: 2009-07-14

Version: 1.0.0

Record Log Profile SM CLP Mapping Specification

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

CONTENTS

Tables

60

Foreword

The *Record Log Profile SM CLP Mapping Specification* (DSP0810) 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 Inc.
- Jon Hass – Dell, Inc.
- Jeff Hilland – HP
- Christina Shaw – HP
- Aaron Merkin – IBM
- Jeff Lynch – IBM
- Perry Vincent – Intel
- John Leung – Intel

Introduction

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

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

Record Log 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 *Record Log 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 DSP1010, *Record Log Profile1.0*,
http://www.dmtf.org/stadards/published_documents/DSP1010_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

158 **4.5**
159 **SM**
160 Server Management

161 **4.6**
162 **SMI-S**
163 Storage Management Initiative Specification

164 **4.7**
165 **SNIA**
166 Storage Networking Industry Association

167 5 Recipes

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

170 • smResetRSC
171 • smShowInstance
172 • smShowInstances
173 • smShowAssociationInstance
174 • smShowAssociationInstances
175 • smStartRSC
176 • smStopRSC

177 6 Mappings

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

181 6.1 CIM_ElementCapabilities

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

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

188 **Table 1 – 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	

Command Verb	Requirement	Comments
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_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.

6.1.2 Show

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

6.1.2.1 Show Command Form for Multiple Instances Target – CIM_EnabledLogicalElementCapabilities Reference

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

6.1.2.1.1 Command Form

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

6.1.2.1.2 CIM Requirements

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

6.1.2.1.3 Behavior Requirements

6.1.2.1.3.1 Preconditions

In this section `$instance` represents the instance of CIM_EnabledLogicalElementCapabilities which is referenced by CIM_ElementCapabilities.

`#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)>;
}
```

```
&smShowAssociationInstances ( “CIM_ElementCapabilities”, $instance.getInstancePath(),
   #propertylist[] );
&smEnd;
```

6.1.2.2 Show Command Form for a Single Instance – CIM_RecordLog Reference

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

6.1.2.2.1 Command Form

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

6.1.2.2.2 CIM Requirements

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

6.1.2.2.3 Behavior Requirements

6.1.2.2.3.1 Preconditions

In this section `$instance` represents the instance of CIM_RecordLog which is referenced by CIM_ElementCapabilities.

`#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)>;
}
&smShowAssociationInstances ( “CIM_ElementCapabilities”, $instance.getInstancePath(),
   #propertylist[] );
&smEnd;
```

6.1.2.3 Show Command Form for a Single Instance Target – Both References

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

6.1.2.3.1 Command Form

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

6.1.2.3.2 CIM Requirements

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

6.1.2.3.3 Behavior Requirements

6.1.2.3.3.1 Preconditions

In this section `$instanceA` represents the referenced instance of CIM_RecordLog through the CIM_ElementCapabilities association. `$instanceB` represents the instance of CIM_EnabledLogicalElementCapabilities which is referenced by CIM_ElementCapabilities.

`#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)>;
}
&smShowAssociationInstance ( "CIM_ElementCapabilities", $instanceA.getInstancePath(),
   $instanceB.getInstancePath(), #propertylist[] );
&smEnd;
```

6.2 CIM_EnabledLogicalElementCapabilities

The `cd`, `exit`, `help` and `version` 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_EnabledLogicalElementCapabilities

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.2.2.
Start	Not supported	
Stop	Not supported	

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

6.2.1 Ordering of Results

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

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

6.2.2 Show

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

6.2.2.1 Show Command Form for Multiple Instances Target

This command form is used to show many instances of CIM_EnabledLogicalElementCapabilities.

6.2.2.1.1 Command Form

```
show <CIM_EnabledLogicalElementCapabilities multiple instances>
```

6.2.2.1.2 CIM Requirements

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

6.2.2.1.3 Behavior Requirements

6.2.2.1.3.1 Preconditions

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

`#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)>;
}
&smShowInstances ( “CIM_EnabledLogicalElementCapabilities”, “CIM_MemberOfCollection”,
   $containerInstance.getInstancePath(), #propertylist[] );
&smEnd;
```

6.2.2.2 Show Command Form for a Single Instance Target

This command form is used to show a single instance of CIM_EnabledLogicalElementCapabilities.

6.2.2.2.1 Command Form

```
show <CIM_EnabledLogicalElementCapabilities single instance>
```

6.2.2.2.2 CIM Requirements

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

6.2.2.2.3 Behavior Requirements

6.2.2.2.3.1 Preconditions

In this section `$instance` represents the targeted instance of CIM_EnabledLogicalElementCapabilities.

```
$instance=<CIM_EnabledLogicalElementCapabilities single instance>;
```

`#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)>;
}
&smShowInstance ( $instance.getInstancePath(), #propertylist[] );
&smEnd;
```

6.3 CIM_LogManagesRecord

The `cd`, `exit`, `help`, and `version` 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_LogManagesRecord

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

6.3.1 Ordering of Results

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

- Results for CIM_LogManagesRecord 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_LogManagesRecord. Implementations shall support the use of the `show` verb with CIM_LogManagesRecord.

6.3.2.1 Show Command Form for Multiple Instances Target – CIM_RecordLog Reference

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

6.3.2.1.1 Command Form

```
show <CIM_LogManagesRecord multiple instances>
```

6.3.2.1.2 CIM Requirements

See CIM_LogManagesRecord in the “CIM Elements” section of the *Record Log Profile* for the list of mandatory properties.

6.3.2.1.3 Behavior Requirements

6.3.2.1.3.1 Preconditions

In this section $instance represents the instance of CIM_RecordLog which is referenced by CIM_LogManagesRecord.

`#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_LogManagesRecord”, $instance.getInstancePath(),
   #propertylist[] );
&smEnd;
```

6.3.2.2 Show Command Form for a Single Instance – CIM_LogEntry Reference

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

6.3.2.2.1 Command Form

```
show <CIM_LogManagesRecord single instance>
```

6.3.2.2.2 CIM Requirements

See CIM_LogManagesRecord in the “CIM Elements” section of the *Record Log Profile* for the list of mandatory properties.

6.3.2.2.3 Behavior Requirements

6.3.2.2.3.1 Preconditions

In this section `$instance` represents the instance of CIM_LogEntry which is referenced by CIM_LogManagesRecord.

`#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_LogManagesRecord”, $instance.getInstancePath(),
   #propertylist[] );
&smEnd;
```

6.3.2.3 Show Command Form for a Single Instance Target – Both References

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

6.3.2.3.1 Command Form

```
show <CIM_LogManagesRecord single instance>
```

6.3.2.3.2 CIM Requirements

See CIM_LogManagesRecord in the “CIM Elements” section of the *Record Log Profile* for the list of mandatory properties.

6.3.2.3.3 Behavior Requirements

6.3.2.3.3.1 Preconditions

In this section `$instanceA` represents the referenced instance of CIM_RecordLog through CIM_LogManagesRecord association. `$instanceB` represents the instance of CIM_LogEntry which is referenced by CIM_LogManagesRecord.

`#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_LogManagesRecord”, $instanceA.getInstancePath(),
   $instanceB.getInstancePath(), #propertylist[] );
&smEnd;
```

6.4 CIM_LogEntry

The `cd`, `exit`, `help` and `version` 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_LogEntry

Command Verb	Requirement	Comments
Create	Not supported	
Delete	May	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.4.3.
Start	Not supported	
Stop	Not supported	

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

6.4.1 Ordering of Results

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

- Results for CIM_LogEntry instances are based on the TimeStamp property of the CIM_LogEntry class. The natural order is descending: LIFO (Last In, First Out), from the newest log entry to the oldest. In the rare case when TimeStamp values for some CIM_LogEntry instances are equal, it is left up to the implementation to sort those instances.

6.4.2 Delete

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

of CIM_LogEntry aggregated within a CIM_RecordLog instance corresponds to clearing the log. Deleting a single instance of CIM_LogEntry corresponds to removing an entry from the log. Implementation may support one or both of the below command forms.

6.4.2.1 Delete Multiple Instances – All Instance of Log Entries

Used to clear all the log entries as all CIM_LogEntry instances contained within the CIM_RecordLog are the target.

6.4.2.1.1 Command Form

```
delete <CIM_LogEntry multiple instances>
```

6.4.2.1.2 CIM Requirements

```
uint32 CIM_RecordLog.ClearLog()
```

6.4.2.1.3 Behavior Requirements

6.4.2.1.3.1 Preconditions

In this section `$containerInstance` is the instance of CIM_RecordLog, which represents the container log for the entries to be deleted, and is associated to the targeted instances of CIM_LogEntry through the CIM_LogManagesRecord association.

6.4.2.1.3.2 Pseudo Code

```
#Error = &smOpEnumerateInstances ( “CIM_LogEntry”, $deleteInstances[] );
if (0 != #Error.code)
{
   //includes &smEnd;
   &smProcessOpError (#Error);
}
%InArguments[] = {};
%OutArguments[] = {};
#Error = InvokeMethod ($containerInstance.getInstancePath(),
      “ClearLog”,
      %InArguments[],
      %OutArguments[],
      #returnStatus);
if (0 != #Error.code)
{
   //method invocation failed
   if ( (null != #Error.$error) && (null != #Error.$error[0]) )
   {
      //if the method invocation contains an embedded error
      //use it for the Error for the overall job
      &smAddError($job, #Error.$error[0]);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else if ( 17 == #Error.code )
   {
      //17 – CIM_ERR_METHOD_NOT_FOUND
      // The specified extrinsic method does not exist.
      $OperationError = smNewInstance(“CIM_Error”);
```

```
      // CIM_ERR_METHOD_NOT_FOUND
      $OperationError.CIMStatusCode = 17;
      //Software Error
      $OperationError.ErrorType = 10;
      //Unknown
      $OperationError.PerceivedSeverity = 0;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000001;
      $OperationError.Message = “Operation is not supported.”
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else
   {
      //operation failed, but no detailed error instance, need to make one up
      //make an Error instance and associate with job for Operation
      $OperationError = smNewInstance(“CIM_Error”);
      //CIM_ERR_FAILED
      $OperationError.CIMStatusCode = 1;
      //Software Error
      $OperationError.ErrorType = 4;
      //Unknown
      $OperationError.PerceivedSeverity = 0;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000009;
      $OperationError.Message = “An internal software error has occurred.”;
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   }
}
//if CIM op failed
else if (0 == #returnStatus)
{
   //completed successfully
   &smCommandCompleted($job);
   &smEnd;
}
else if (1 == #returnStatus)
{
   //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
{
   // generic failure
   $OperationError = smNewInstance(“CIM_Error”);
   //CIM_ERR_FAILED
   $OperationError.CIMStatusCode = 1;
   //Other
   $OperationError.ErrorType = 1;
   //Low
   $OperationError.PerceivedSeverity = 2;
   $OperationError.OwningEntity = DMTF:SMCLP;
   $OperationError.MessageID = 0x00000002;
   $OperationError.Message = “Failed.  No further information is available.”;
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
   &smEnd;
}
for #i in $deleteInstances[]
{
   &smDisplayInstance ( $deleteInstances[#i] );
}
&smEnd;
```

6.4.2.2 Delete a Single Instance

This command is used to delete a single entry within the log.

6.4.2.2.1 Command Form

```
delete <CIM_LogEntry single instance>
```

6.4.2.2.2 CIM Requirements

See CIM_LogEntry in the “CIM Elements” section of the *Record Log Profile* for the list of mandatory properties.

6.4.2.2.3 Behavior Requirements

6.4.2.2.3.1 Preconditions

```
$instanceLogRecord=<CIM_LogEntry single instance>;
```

6.4.2.2.3.2 Pseudo Code

```
#Error = &smOpDeleteInstance ( $instanceLogRecord.getInstancePath() );
//deletes association classes referencing the instance as well
if (0 != Error.code)
{
   &smProcessOpError (#Error);
   //includes end;
}
&smDisplayInstance ( $instanceLogRecord );
&smEnd;
```

6.4.3 Show

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

6.4.3.1 Show Command Form for Multiple Instances Target

This command form is used to show many instances of CIM_LogEntry.

6.4.3.1.1 Command Form

```
show <CIM_LogEntry multiple instances>
```

6.4.3.1.2 CIM Requirements

See CIM_LogEntry in the “CIM Elements” section of the *Record Log Profile* for the list of mandatory properties.

6.4.3.1.3 Behavior Requirements

6.4.3.1.3.1 Preconditions

In this section `$containerInstance` represents the instance of CIM_RecordLog which represents the log and is associated to the targeted instances of CIM_LogEntry through the CIM_LogManagesRecord association.

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

6.4.3.1.3.2 Pseudo Code

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

6.4.3.2 Show Command Form for a Single Instance Target

This command form is used to show a single instance of CIM_LogEntry.

6.4.3.2.1 Command Form

```
show <CIM_LogEntry single instance>
```

6.4.3.2.2 CIM Requirements

See CIM_LogEntry in the “CIM Elements” section of the *Record Log Profile* for the list of mandatory properties.

6.4.3.2.3 Behavior Requirements

6.4.3.2.3.1 Preconditions

In this section `$instance` represents the targeted instance of CIM_LogEntry.

```
$instance=<CIM_LogEntry single instance>;
```

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

6.4.3.2.3.2 Pseudo Code

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

6.5 CIM_RecordLog

The `cd`, `exit`, `help`, and `version` 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_RecordLog

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	May	See 6.5.2.
Set	May	See 6.5.3.
Show	Shall	See 6.5.4.
Start	May	See 6.5.5.
Stop	May	See 6.5.6.

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

6.5.1 Ordering of Results

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

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

6.5.2 Reset

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

6.5.2.1 Command Form

```
reset <CIM_RecordLog single instance>
```

6.5.2.2 CIM Requirements

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

6.5.2.3 Behavior Requirements

6.5.2.3.1.1 Preconditions

In this section `$instance` represents the targeted instance of CIM_RecordLog.

```
$instance=<CIM_RecordLog single instance>;
```

6.5.2.3.2 Pseudo Code

```
&smResetRSC ( $instance.getInstancePath() );
&smEnd;
```

6.5.3 Set

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

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

6.5.3.1 General Usage of Set for a Single Property

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

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

6.5.3.1.1 Command Form

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

6.5.3.1.2 CIM Requirements

See CIM_RecordLog in the “CIM Elements” section of the *Record Log Profile* for the list of mandatory properties.

6.5.3.1.3 Behavior Requirements

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

6.5.3.2 General Usage of Set for Multiple Properties

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

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

6.5.3.2.1 Command Form

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

6.5.3.2.2 CIM Requirements

See CIM_RecordLog in the “CIM Elements” section of the *Record Log Profile* for the list of mandatory properties.

6.5.3.2.3 Behavior Requirements

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

6.5.4 Show

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

6.5.4.1 Show Command Form for Multiple Instances Target

This command form is used to show many instances of CIM_RecordLog.

6.5.4.1.1 Command Form

```
show <CIM_RecordLog multiple instances>
```

6.5.4.1.2 CIM Requirements

See CIM_RecordLog in the “CIM Elements” section of the *Record Log Profile* for the list of mandatory properties.

6.5.4.1.3 Behavior Requirements

6.5.4.1.3.1 Preconditions

In this section `$containerInstance` represents the instance of CIM_ConcreteCollection which represents the collection of logs and is associated to the targeted instances of CIM_RecordLog through the CIM_MemberOfCollection association.

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

6.5.4.1.3.2 Pseudo Code

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

6.5.4.2 Show Command Form for a Single Instance Target

This command form is used to show a single instance of CIM_RecordLog.

6.5.4.2.1 Command Form

```
show <CIM_RecordLog single instance>
```

6.5.4.2.2 CIM Requirements

See CIM_RecordLog in the “CIM Elements” section of the *Record Log Profile* for the list of mandatory properties.

6.5.4.2.3 Behavior Requirements

6.5.4.2.3.1 Preconditions

In this section `$instance` represents the targeted instance of CIM_RecordLog.

```
$instance=<CIM_RecordLog single instance>;
```

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

6.5.4.2.3.2 Pseudo Code

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

6.5.5 Start

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

6.5.5.1 Command Form

```
start <CIM_RecordLog single instance>
```

6.5.5.2 CIM Requirements

```
uint16 EnabledState;
```

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

6.5.5.3 Behavior Requirements

6.5.5.3.1.1 Preconditions

In this section `$instance` represents the targeted instance of CIM_RecordLog.

```
$instance=<CIM_RecordLog single instance>;
```

6.5.5.3.1.2 Pseudo Code

```
&smStartRSC ( $instance.getInstancePath() );
&smEnd;
```

6.5.6 Stop

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

6.5.6.1 Command Form

```
stop <CIM_RecordLog single instance>
```

6.5.6.2 CIM Requirements

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

6.5.6.3 Behavior Requirements

6.5.6.3.1 Preconditions

In this section `$instance` represents the targeted instance of CIM_RecordLog.

```
$instance=<CIM_RecordLog single instance>;
```

6.5.6.3.1.1 Pseudo Code

```
&smStopRSC ( $instance.getInstancePath() );
&smEnd;
```

6.6 CIM_UseOfLog

The `cd`, `exit`, `help`, and `version` 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.

805 **Table 6 – Command Verb Requirements for CIM_UseOfLog**

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	

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

808 ### 6.6.1 Ordering of Results

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

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

812 ### 6.6.2 Show

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

815 #### 6.6.2.1 Show Command Form for Multiple Instances Target – CIM_RecordLog Reference

816 This command form is used to show many instances of CIM_UseOfLog. This command form corresponds
817 to a `show` command issued against instances of CIM_UseOfLog where only one reference is specified
818 and the reference is to the scoping instance of CIM_RecordLog.

819 ##### 6.6.2.1.1 Command Form

820 ```
show <CIM_UseOfLog multiple instances>
```

821 ##### 6.6.2.1.2 CIM Requirements

822 See CIM_UseOfLog in the “CIM Elements” section of the *Record Log Profile* for the list of mandatory
823 properties.

824 ##### 6.6.2.1.3 Behavior Requirements

825 ###### 6.6.2.1.3.1 Preconditions

826 In this section `$instance` represents the instance of CIM_RecordLog which is referenced by
827 CIM_UseOfLog.

828 `#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_UseOfLog”, $instance.getInstancePath(),
   #propertylist[] );
&smEnd;
```

6.6.2.2 Show Command Form for Multiple Instances – CIM_ManagedSystemElement Reference

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

6.6.2.2.1 Command Form

```
show <CIM_UseOfLog multiple instances>
```

6.6.2.2.2 CIM Requirements

See CIM_UseOfLog in the “CIM Elements” section of the *Record Log Profile* for the list of mandatory properties.

6.6.2.2.3 Behavior Requirements

6.6.2.2.3.1 Preconditions

In this section `$instance` represents the instance of a subclass of CIM_ManagedSystemElement which is referenced by CIM_UseOfLog.

`#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_UseOfLog”, $instance.getInstancePath(),
   propertylist[] );
&smEnd;
```

6.6.2.3 Show Command Form for a Single Instance Target – Both References

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

6.6.2.3.1 Command Form

```
show <CIM_UseOfLog single instance>
```

6.6.2.3.2 CIM Requirements

See CIM_UseOfLog in the “CIM Elements” section of the *Record Log Profile* for the list of mandatory properties.

6.6.2.3.3 Behavior Requirements

6.6.2.3.3.1 Preconditions

In this section `$instanceA` represents the referenced instance of CIM_RecordLog through CIM_UseOfLog association. `$instanceB` represents the instance of a subclass of CIM_ManagedSystemElement which is referenced by CIM_UseOfLog.

`#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_UseOfLog”, $instanceA.getInstancePath(),
   $instanceB.getInstancePath(), propertylist[] );
&smEnd;
```

ANNEX A
(informative)

Change Log

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