Document Number: DSP0827

Date: 2009-07-14

Version: 1.0.0

Software Update Profile SM CLP Mapping Specification

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

CONTENTS

Foreword 5
Introduction 6
1 Scope 7
2 Normative References 7
2.1 Approved References 7
2.2 Other References 7
3 Terms and Definitions 7
4 Symbols and Abbreviated Terms 8
5 Recipes 9
5.1 smInstallFromURI 9
5.2 smInstallFromSoftwareIdentity 12
6 Mappings 15
6.1 CIM_SoftwareIdentity 15
6.2 CIM_HostedService 17
6.3 CIM_SoftwareInstallationServiceCapabilities 20
6.4 CIM_ElementCapabilities 22
6.5 CIM_ServiceAffectsElement 25
6.6 CIM_ManagedElement 31
ANNEX A (informative) Change Log 35

Tables

Table 1 – Command Verb Requirements for CIM_SoftwareInstallationService 15
Table 2 – Command Verb Requirements for CIM_HostedService 17
Table 3 – Command Verb Requirements for CIM_SoftwareInstallationServiceCapabilities 20
Table 4 – Command Verb Requirements for CIM_ElementCapabilities 22
Table 5 – Command Verb Requirements for CIM_ServiceAffectsElement 25
Table 6 – Command Verb Requirements for CIM_ManagedElement 31

59

Foreword

The *Software Update Profile SM CLP Mapping Specification* (DSP0827) 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 the SMI-S 1.1.0, 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 Update Profile*. The information in this specification, combined with *SM CLP-to-CIM Common Mapping Specification 1.0* (DSP0216), is intended to be sufficient to implement SM CLP commands relevant to the classes, properties, and methods described in the *Software Update Profile* using CIM operations.

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

Software Update 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 Update 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 DSP1025, *Software Update Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1025_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
SM
Server Management

4.5
SMI-S
Storage Management Initiative Specification

4.6
SNIA
Storage Networking Industry Association

4.7
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
- smNewInstance
- smAddError

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

5.1 smInstallFromURI

5.1.1 Description

This method is used to install software that is located at a URI on a managed element.

5.1.2 Pseudo Code

```
sub void smInstallfromURI (#InstallOptions[],#uri, $targetME->, $targetSWInsSer->)
{
//#InstallOptions contains the list of options that control the installation procedure
//$targetME parameter contains the target managed element on which the software needs
//to be installed. uri parameter contains the URI at which the software is located
$instanceConcreteJob = smNewInstance (“CIM_ConcreteJob”);
%InArguments[] = { newArgument(“InstallOptions[]”, #InstallOptions),
   newArgument(“URI”, #uri), newArgument(“Target”, $targetME->) }
   %OutArguments[] = { newArgument(“Job”, $instanceConcreteJob.getObjectPath() ) }
   #Error = smOpInvokeMethod ($targetSWInsSer->,
      “InstallFromURI”,
      %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 == #returnStatus) {
    //The specified extrinsic method does not exist
$OperationError = smNewInstance("CIM_Error");
//CIM_ERR_METHOD_NOT_FOUND
$OperationError.CIMStatusCode = 17;
//Software Error
$OperationError.ErrorType = 10;
//Low
$OperationError.PerceivedSeverity = 0;
$OperationError.OwningEntity = DMTF:SMCLP;
          $OperationError.MessageID = 0x00000001;
          $OperationError.Message = "Operation is not supported";
          &smAddError($job, $OperationError);
&smMakeCommandStatus($job);
}
          else
          {
//operation failed, but no detailed error instance, need to make //one up
             //make an Error instance and associate with job for Operation
             $OperationError = smNewInstance("CIM_Error");
             //CIM_ERR_FAILED
             $OperationError.CIMStatusCode = 1;
             //Software Error
             $OperationError.ErrorType = 4;
             //Unknown
             $OperationError.PerceivedSeverity = 0;
             $OperationError.OwningEntity = DMTF:SMCLP;
             $OperationError.MessageID = 0x00000009;
$OperationError.Message = "An internal software error has occurred.";
             &smAddError($job, $OperationError);
             &smMakeCommandStatus($job);
             &smEnd;
          }
}//if CIM op failed
else if (0 == #returnStatus)  {
   //completed successfully
   &smCommandCompleted($job);
   &smEnd;
}
else if (4096 == #returnStatus)   {
//job spawned, need to watch for it to finish
//while the jobstate is 4 ("Running")
while (4 == $instanceConcreteJob.JobState){<busy wait>}
//when job finishes, invoke GetError()
```

```
if (2 != $job.OperationalStatus) {
%InArguments[] = { }
%OutArguments[] = {newArgument(“Job”, $instanceConcreteJob.getObjectPath())}
#Error = smOpInvokeMethod($job, “GetError”, %InArguments, %OutArguments, #returncode);
//Method invocation failed, internal processing error
if ( (0 != #Error.code) || (0 != #returncode) )  {
//make an Error instance and associate with job for Operation
$OperationError = smNewInstance(“CIM_Error”);
//CIM_ERR_FAILED
$OperationError.CIMStatusCode = 1;
//Software Error
$OperationError.ErrorType = 4;
//Unknown
$OperationError.PerceivedSeverity = 0;
$OperationError.OwningEntity = DMTF:SMCLP;
$OperationError.MessageID = 0x00000009;
$OperationError.Message = “An internal software error has occurred.”;
&smAddError($job, $OperationError);
&smMakeCommandStatus($job);
&smEnd;
} else {
//make command status
$joberror = %OutArguments[“Error”];
&smMakeCommandExecutionFailed($job, {$joberror};
} //end if have CIM_Error from GetError()
}
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 if (2 == #returnStatus)  {
   //generic failure
$OperationError = smNewInstance(“CIM_Error”);
//CIM_ERR_FAILED
$OperationError.CIMStatusCode = 1;
//Other
$OperationError.ErrorType = 1;
//Low
$OperationError.PerceivedSeverity = 2;
$OperationError.OwningEntity = DMTF:SMCLP;
$OperationError.MessageID = 0x00000002;
```

```
          $OperationError.Message = “Failed. No further information is available.”;
          &smAddError($job, $OperationError);
          &smMakeCommandStatus($job);
}
else   {
   //unspecified return code, generic failure
$OperationError = smNewInstance(“CIM_Error”);
//CIM_ERR_FAILED
$OperationError.CIMStatusCode = 1;
//Other
$OperationError.ErrorType = 1;
//Low
$OperationError.PerceivedSeverity = 2;
$OperationError.OwningEntity = DMTF:SMCLP;
$OperationError.MessageID = 0x00000002;
$OperationError.Message = “Failed. No further information is available.”;
&smAddError($job, $OperationError);
&smMakeCommandStatus($job);
&smEnd;
}
```

5.2 smInstallFromSoftwareIdentity

5.2.1 Description

This method is used to install software, represented by an instance of CIM_SoftwareIdentity, on a managed element.

5.2.2 Pseudo Code

```
sub void smInstallFromSoftwareIdentity (#InstallOptions[], $SWInstance->, $targetME->,
   $targetSWInsSer->)
{
//InstallOptions[] contains the list of options that control the installation
procedure
// $SWInstance represents the instance of CIM_SoftwareIdentity
//$targetME parameter contains the target managed element on which the software needs
to be installed.
$instanceConcreteJob = smNewInstance (”CIM_ConcreteJob”);
%InArguments[] = {newArgument(“InstallOptions[]”, #InstallOptions),
newArgument(“Source”, $SWInstance->),
newArgument(“Target”, $targetME->)}
%OutArguments[] = {newArgument(“Job”, $instanceConcreteJob.getObjectPath())}
#Error = smOpInvokeMethod ($targetSWInsSer->,
”InstallFromSoftwareIdentity”,
%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 == #returnStatus) {
   //The specified extrinsic method does not exist
$OperationError = smNewInstance("CIM_Error");
//CIM_ERR_METHOD_NOT_FOUND
$OperationError.CIMStatusCode = 17;
//Software Error
$OperationError.ErrorType = 10;
//Low
$OperationError.PerceivedSeverity = 0;
$OperationError.OwningEntity = DMTF:SMCLP;
          $OperationError.MessageID = 0x00000001;
          $OperationError.Message = "Operation is not supported";
          &smAddError($job, $OperationError);
          &smMakeCommandStatus($job);
          }
          else
          {
//operation failed, but no detailed error instance, need to make //one up
//make an Error instance and associate with job for Operation
          $OperationError = smNewInstance("CIM_Error");
          //CIM_ERR_FAILED
          $OperationError.CIMStatusCode = 1;
          //Software Error
          $OperationError.ErrorType = 4;
          //Unknown
          $OperationError.PerceivedSeverity = 0;
          $OperationError.OwningEntity = DMTF:SMCLP;
          $OperationError.MessageID = 0x00000009;
$OperationError.Message = "An internal software error has occurred.";
          &smAddError($job, $OperationError);
          &smMakeCommandStatus($job);
          &smEnd;
          }
}//if CIM op failed
else if (0 == #returnStatus)  {
   //completed successfully
   &smCommandCompleted($job);
   &smEnd;
}
else if (4096 == #returnStatus)   {
//job spawned, need to watch for it to finish
//while the jobstate is 4 ("Running")
while (4 == $instanceConcreteJob.JobState){<busy wait>}
//when job finishes, invoke GetError()
if (2 != $job.OperationalStatus) {
%InArguments[] = { }
%OutArguments[] = {newArgument("Job", $instanceConcreteJob.getObjectPath())}
#Error = smOpInvokeMethod($job, "GetError", %InArguments, %OutArguments, #returncode);
```

```
//Method invocation failed, internal processing error
if ( (0 != #Error.code) || (0 != #returncode) )  {
//make an Error instance and associate with job for Operation
$OperationError = smNewInstance(“CIM_Error”);
//CIM_ERR_FAILED
$OperationError.CIMStatusCode = 1;
//Software Error
$OperationError.ErrorType = 4;
//Unknown
$OperationError.PerceivedSeverity = 0;
$OperationError.OwningEntity = DMTF:SMCLP;
$OperationError.MessageID = 0x00000009;
$OperationError.Message = “An internal software error has occurred.”;
&smAddError($job, $OperationError);
&smMakeCommandStatus($job);
&smEnd;
} else {
//make command status
$joberror = %OutArguments[“Error”];
&smMakeCommandExecutionFailed($job, {$joberror};
} //end if have CIM_Error from GetError()
}
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 if (2 == #returnStatus)  {
   //generic failure
$OperationError = smNewInstance(“CIM_Error”);
//CIM_ERR_FAILED
$OperationError.CIMStatusCode = 1;
//Other
$OperationError.ErrorType = 1;
//Low
$OperationError.PerceivedSeverity = 2;
$OperationError.OwningEntity = DMTF:SMCLP;
$OperationError.MessageID = 0x00000002;
   $OperationError.Message = “Failed. No further information is available.”;
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
}
```

```
else   {
   //unspecified return code, generic failure
$OperationError = smNewInstance("CIM_Error");
//CIM_ERR_FAILED
$OperationError.CIMStatusCode = 1;
//Other
$OperationError.ErrorType = 1;
//Low
$OperationError.PerceivedSeverity = 2;
$OperationError.OwningEntity = DMTF:SMCLP;
$OperationError.MessageID = 0x00000002;
$OperationError.Message = "Failed. No further information is available.";
&smAddError($job, $OperationError);
&smMakeCommandStatus($job);
&smEnd;
}
```

6 Mappings

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

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

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

6.1.1 Ordering of Results

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

- Results for CIM_SoftwareInstallationService 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_SoftwareInstallationService. Implementations shall support the use of the `show` verb with CIM_SoftwareInstallationService.

6.1.2.1 Show Command Form for Single Object Target – CIM_SoftwareInstallationService

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

6.1.2.1.1 Command Form

```
show <CIM_SoftwareInstallationService single object>
```

6.1.2.1.2 CIM Requirements

See CIM_SoftwareInstallationService in the “CIM Elements” section of the *Software Update 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_SoftwareInstallationService.

```
$instance = <CIM_SoftwareInstallationService 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 – CIM_SoftwareInstallationService

6.1.2.2.1 Command Form

```
show <CIM_SoftwareInstallationService multiple objects>
```

6.1.2.2.2 CIM Requirements

See CIM_SoftwareInstallationService in the “CIM Elements” section of the *Software Update 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_SoftwareInstallationSerivce through the CIM_HostedService 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_SoftwareInstallationService”, “CIM_HostedService”,
   $containerInstance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.2 CIM_HostedService

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_HostedService

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

Command Verb	Requirement	Comments
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_HostedService, implementations shall utilize the following algorithm to produce the natural (that is, default) ordering:

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

6.2.2 Show

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

6.2.2.1 Show Command Form for Multiple Instances Target – CIM_ComputerSystem Reference

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

6.2.2.1.1 Command Form

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

6.2.2.1.2 CIM Requirements

See CIM_HostedService in the “CIM Elements” section of the *Software Update 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_ComputerSystem, which is referenced by CIM_HostedService.

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

`#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_HostedService”, $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.2.2.2 Show Command Form for a Single Instance – CIM_SoftwareInstallationService Reference

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

6.2.2.2.1 Command Form

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

6.2.2.2.2 CIM Requirements

See CIM_HostedService in the “CIM Elements” section of the *Software Update 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_SoftwareInstallationService, which is referenced by CIM_HostedService.

```
$instance=<CIM_SoftwareInstallationService 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)>;
   }
&smShowAssociationInstances ( “CIM_HostedService”, $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.2.2.3 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 the `show` command issued against CIM_HostedService where both references are specified and therefore the desired instance is unambiguously identified.

6.2.2.3.1 Command Form

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

6.2.2.3.2 CIM Requirements

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

6.2.2.3.3 Behavior Requirements

6.2.2.3.3.1 Preconditions

In this section `$instanceA` represents the instance of a CIM_ComputerSystem and `$instanceB` represents the instance of CIM_SoftwareInstallationService, both of which are referenced by CIM_HostedService.

```
$instanceA=<CIM_ComputerSystem single instance>;
$instanceB=<CIM_SoftwareInstallationService single instance>;
```

`#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_HostedService", $instanceA.getObjectPath(),
   $instanceB.getObjectPath(), #propertylist[] );
&smEnd;
```

6.3 CIM_SoftwareInstallationServiceCapabilities

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_SoftwareInstallationServiceCapabilities

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.3.2.
start	Not supported	
stop	Not supported	

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

6.3.1 Ordering of Results

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

- Results for CIM_SoftwareInstallationServiceCapabilities 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_SoftwareInstallationServiceCapabilities. Implementations shall support the use of the `show` verb with CIM_SoftwareInstallationServiceCapabilities.

The `show` verb is used to display information about CIM_SoftwareInstallationServiceCapabilities instances.

6.3.2.1 Show Command Form for a Single Instance Target

6.3.2.1.1 Command Form

```
show <CIM_SoftwareInstallationServiceCapabilities single instance>
```

6.3.2.1.2 CIM Requirements

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

6.3.2.1.3 Behavior Requirements

6.3.2.1.3.1 Preconditions

`$instance` represents the targeted instance of CIM_SoftwareInstallationServiceCapabilities.

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

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

6.3.2.2 Show Command Form for Multiple Instances Target

6.3.2.2.1 Command Form

```
show <CIM_SoftwareInstallationServiceCapabilities multiple instances>
```

6.3.2.2.2 CIM Requirements

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

6.3.2.2.3 Behavior Requirements

6.3.2.2.3.1 Preconditions

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

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

6.4 CIM_ElementCapabilities

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_ElementCapabilities

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

Command Verb	Requirement	Comments
start	Not supported	
stop	Not supported	

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

6.4.1 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.

6.4.2 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.

6.4.2.1 Show Command Form for a Single Instance Target – CIM_SoftwareInstallationService Reference

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

6.4.2.1.1 Command Form

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

6.4.2.1.2 CIM Requirements

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

6.4.2.1.3 Behavior Requirements

6.4.2.1.3.1 Preconditions

`$instance` represents the instance of a CIM_SoftwareInstallationService, which is referenced by CIM_ElementCapabilities.

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

`#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)>;
   }
&smShowAssociationInstances ( “CIM_ElementCapabilities”, $instance.getObjectPath(),
```

```
   #propertylist[] );
&smEnd;
```

6.4.2.2 Show Command Form for Multiple Instances – CIM_SoftwareInstallationServiceCapabilities Reference

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

6.4.2.2.1 Command Form

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

6.4.2.2.2 CIM Requirements

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

6.4.2.2.3 Behavior Requirements

6.4.2.2.3.1 Preconditions

`$instance` represents the instance of a CIM_SoftwareInstallationServiceCapabilities, which is referenced by CIM_ElementCapabilities.

```
$instance=<CIM_SoftwareInstallationServiceCapabilities single instance>.
```

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

6.4.2.3 Show Command Form for a Single Instance Target – CIM_SoftwareInstallationService and CIM_SoftwareInstallationServiceCapabilities 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_ElementCapabilities where both references are specified and therefore the desired instance is unambiguously identified.

6.4.2.3.1 Command Form

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

6.4.2.3.2 CIM Requirements

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

6.4.2.3.3 Behavior Requirements

6.4.2.3.3.1 Preconditions

`$instanceA` represents the instance of a CIM_SoftwareInstallationService and `$instanceB` represents the instance of a CIM_SoftwareInstallationServiceCapabilities, both of which are referenced by CIM_ElementCapabilities.

```
$instanceA=<CIM_SoftwareInstallationService single instance>;
$instanceB=<CIM_SoftwareInstallationServiceCapabilities single instance>;
```

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

6.5 CIM_ServiceAffectsElement

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_ServiceAffectsElement

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

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

6.5.1 Ordering of Results

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

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

6.5.2 Show

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

6.5.2.1 Show Command Form for Multiple Instances Target – CIM_SoftwareInstallationService Reference

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

6.5.2.1.1 Command Form

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

6.5.2.1.2 CIM Requirements

See CIM_ServiceAffectsElement in the “CIM Elements” section of the *Software Update 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 a CIM_SoftwareInstallationService, which is referenced by CIM_ServiceAffectsElement.

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

`#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_ServiceAffectsElement”, $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.5.2.2 Show Command Form for Multiple Instances – CIM_SoftwareIdentity Reference

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

6.5.2.2.1 Command Form

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

6.5.2.2.2 CIM Requirements

See CIM_ServiceAffectsElement in the “CIM Elements” section of the *Software Update 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_SoftwareIdentity, which is referenced by CIM_ServiceAffectsElement.

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

`#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_ServiceAffectsElement”, $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.5.2.3 Show Command Form for a Single Instance Target – CIM_SoftwareInstallationService and CIM_SoftwareIdentity References

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

6.5.2.3.1 Command Form

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

6.5.2.3.2 CIM Requirements

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

6.5.2.3.3 Behavior Requirements

6.5.2.3.3.1 Preconditions

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

```
$instanceA=<CIM_SoftwareInstallationService single instance>;
$instanceB=<CIM_SoftwareIdentity single instance>;
```

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

6.5.2.4 Show Command Form for Multiple Instances – CIM_ManagedElement Reference

This command form is used to show a single instance of CIM_ServiceAffectsElement. This command form corresponds to a show command issued against multiple instances of CIM_ServiceAffectsElement where only one reference is specified and the reference is to the instance of CIM_ManagedElement.

6.5.2.4.1 Command Form

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

6.5.2.4.2 CIM Requirements

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

6.5.2.4.3 Behavior Requirements

6.5.2.4.3.1 Preconditions

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

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

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

6.5.2.4.3.2 Pseudo Code

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

6.5.2.5 Show Command Form for a Single Instance Target – CIM_SoftwareInstallationService and CIM_ManagedElement References

This command form is for the show verb applied to a single instance. This command form corresponds to the show command issued against CIM_ServiceAffectsElement where both references of

CIM_SoftwareInstallationService and CIM_ManagedElement are specified and therefore the desired instance is unambiguously identified.

6.5.2.5.1 Command Form

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

6.5.2.5.2 CIM Requirements

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

6.5.2.5.3 Behavior Requirements

6.5.2.5.3.1 Preconditions

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

```
$instanceA=<CIM_SoftwareInstallationService single instance>;
$instanceB=<CIM_ManagedElement single instance>;
```

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

6.5.2.5.3.2 Pseudo Code

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

6.5.2.6 Show Command Form for Multiple Instances – CIM_ComputerSystem Reference

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

6.5.2.6.1 Command Form

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

6.5.2.6.2 CIM Requirements

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

6.5.2.6.3 Behavior Requirements

6.5.2.6.3.1 Preconditions

`$instance` represents the instance of a CIM_ComputerSystem, which is referenced by CIM_ServiceAffectsElement.

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

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

6.5.2.6.3.2 Pseudo Code

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

6.5.2.7 Show Command Form for a Single Instance Target – CIM_SoftwareInstallationService and CIM_ComputerSystem References

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

6.5.2.7.1 Command Form

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

6.5.2.7.2 CIM Requirements

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

6.5.2.7.3 Behavior Requirements

6.5.2.7.3.1 Preconditions

$instanceA represents the instance of a CIM_SoftwareInstallationService and $instanceB represents the instance of a CIM_ComputerSystem, both of which are referenced by CIM_ServiceAffectsElement.

```
$instanceA=<CIM_SoftwareInstallationService single instance>;
$instanceB=<CIM_ComputerSystem single instance>;
```

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

6.5.2.7.3.2 Pseudo Code

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

6.6 CIM_ManagedElement

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_ManagedElement

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Shall	See 6.6.1.
reset	Not supported	
set	Not supported	
show	Not supported	
start	Not supported	
stop	Not supported	

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

6.6.1 Load

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

6.6.1.1 Load Command Form to Install Software Using a URI

This section describes how to implement the `load` verb that is used to install or update software on an instance of CIM_ManagedElement. The location of the software is represented by a URI. When the Install, Update, Repair, Reboot, Uninstall, or SilentMode property is specified, the format is expected to be in Boolean format as defined in DSP0216. When the Password, Log, or AdministrativeMode is specified, the format is expected to be in String format as defined in DSP0216.

6.6.1.1.1 Command Form

```
load –source <URI>
   Install=<Install option value 1>
   Update=<Install option value 2>
   Repair=<Install option value 3>
   Reboot=<Install option value 4>
   Password=<Install option value 5>
   Uninstall=<Install option value 6>
   Log=<Install option value 7>
   SilentMode=<Install option value 8>
   AdministrativeMode=<Install option value 9> <CIM_ManagedElement single instance>
```

6.6.1.1.2 CIM Requirements

```
CIM_SoftwareInstallationService.InstallFromURI
```

6.6.1.1.3 Behavior Requirements

```
$MEinstance=<CIM_ManagedElement single instance>
$uri = <URI>
for #i < n
   {
   #InstallOptions[#i] = <Install option value #i>
   }
#Error = &smOpAssociators(
 $instance.getObjectPath(),
“CIM_ServiceAffectsElement”,
“CIM_SoftwareInstallationService”,
NULL,
NULL,
NULL,
 $SWInsSerInstancePaths[])
if (0 != #Error.code)
{
&smProcessOpError (#Error);
//includes &smEnd;
}
else if (SWInsSerInstancePaths.length() > 0)
{
for #i < n //n is the number of CIM_SoftwareInstallationService instances
 {
       $SWInsSerinstance = $SWInsSerInstancePaths[i];
       #Error= smInstallFromURI(InstallOptions[], uri, MEinstance->,
                    SWInsSerinstance->);
if (0 == #Error.code)
{
#i = n
//goto &smEnd;
}
 }
}
else
{
//unspecified return code, generic failure
$OperationError = smNewInstance(“CIM_Error”);
//CIM_ERR_FAILED
$OperationError.CIMStatusCode = 1;
//Other
$OperationError.ErrorType = 1;
//Low
$OperationError.PerceivedSeverity = 2;
$OperationError.OwningEntity = DMTF:SMCLP;
$OperationError.MessageID = 0x00000001;
$OperationError.Message = “Operation is not supported”;
&smAddError($job, $OperationError);
&smMakeCommandStatus($job);
}
&smEnd;
```

6.6.1.2 Load Command Form to Install Software Using a Software Identity

This section describes how to implement the `load` verb that is used to install or update an instance of CIM_SoftwareIdentity on an instance of CIM_ManagedElement.

When the Install, Update, Repair, Reboot, Uninstall, or SilentMode property is specified, the format is expected to be in Boolean format as defined in DSP0216. When the Password, Log, or AdministrativeMode is specified, the format is expected to be in String format as defined in DSP0216.

6.6.1.2.1 Command Form

```
load –source <CIM_SoftwareIdentity single instance>
   Install=<Install option value 1>
   Update=<Install option value 2>
   Repair=<Install option value 3>
   Reboot=<Install option value 4>
   Password=<Install option value 5>
   Uninstall=<Install option value 6>
   Log=<Install option value 7>
   SilentMode=<Install option value 8>
   AdministrativeMode=<Install option value 9>
   <CIM_ManagedElement single instance>
```

6.6.1.2.2 CIM Requirements

```
CIM_SoftwareInstallationService.InstallFromSoftwareIdentity
```

6.6.1.2.3 Behavior Requirements

```
$MEinstance=<CIM_ManagedElement single instance>
$SWinstance=<CIM_SoftwareIdentity single instance>
for #i < n
   {
      #InstallOptions[#i] = <Install option value #i>
   }
#Error = &smOpAssociators (
         $instance.getObjectPath(),
         “CIM_ServiceAffectsElement”,
         “CIM_SoftwareInstallationService”,
         NULL,
         NULL,
         NULL,
         $SWInsSerInstancePaths[] )
if (0 != #Error.code)
   {
      &smProcessOpError (#Error);
      //includes &smEnd;
   }
else if ( SWInsSerInstancePaths.length() > 0 )
   {
   for #i < n //n is the number of CIM_SoftwareInstallationService instances
   {
   $SWInsSerinstance = $SWInsSerInstancePaths[i];
   #Error=smInstallFromSoftwareIdentity ( #InstallOptions[],
   $SWinstance.getObjectPath(), $MEinstance.getObjectPath(),
   $SWInsSerinstance.getObjectPath() );
if ( 0 == #Error.code )
   {
      #i = n
```

```
       //goto &smEnd;
   }
else
   {
       //unspecified return code, generic failure
       $OperationError = smNewInstance(“CIM_Error”);
       //CIM_ERR_FAILED
       $OperationError.CIMStatusCode = 1;
       //Other
       $OperationError.ErrorType = 1;
       //Low
       $OperationError.PerceivedSeverity = 2;
       $OperationError.OwningEntity = DMTF:SMCLP;
       $OperationError.MessageID = 0x00000001;
       $OperationError.Message = “Operation is not supported”;
       &smAddError($job, $OperationError);
       &smMakeCommandStatus($job);
   }
&smEnd;
```

ANNEX A
(informative)

Change Log

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