Document Number: DSP1080

Date: 2014-01-14

Version: 2.0.0a

Enabled Logical Element Profile

IMPORTANT: This specification is not a standard. It does not necessarily reflect the views of the DMTF or all of its members. Because this document is a Work in Progress, this specification may still change, perhaps profoundly. This document is available for public review and comment until the stated expiration date.

This document expires on: **2014-06-30**.

Target version for DMTF Standard: **2.0.0**.

Provide any comments through the DMTF Feedback Portal: http://www.dmtf.org/standards/feedback

Document Type: Specification

Document Status: Work in Progress

Document Language: en-US

Copyright notice

CONTENTS

20

Figures

21

Tables

22

Foreword

This document was prepared by the DMTF Architecture Working Group

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. For information about the DMTF, see http://www.dmtf.org.

Acknowledgements

DMTF acknowledges the following individuals for their contributions to this document:

- Andreas Maier, IBM (Editor)
- Jon Hass, Dell Inc. (Editor of V1.0.0)
- Khachatur Papanyan, Dell Inc. (Editor of V1.0.0)
- Barb Craig, HP
- George Ericson, EMC
- Steve Hand, Symantec
- Jeff Hilland, HP
- David Hines, Intel
- Joe Kozlowski, Dell Inc.
- John Leung, Intel
- Aaron Merkin, IBM
- Christina Shaw, HP

Introduction

39 The information in this specification and referenced specifications should be sufficient for a provider or consumer of this data to identify unambiguously the classes, properties, methods, and values that shall be instantiated and manipulated to represent and manage the common aspects of enabled logical elements that are modeled using the DMTF CIM core and extended model definitions.

40 The target audience for this specification is implementers who are writing CIM-based providers or consumers of management interfaces that represent the components described in this document.

41

Document conventions

42

Typographical conventions

43 The following typographical conventions are used in this document:

44 • Document titles are marked in *italics*.

45 • Important terms that are used for the first time are marked in *italics*.

46 • Terms include a link to the term definition in the "Terms and definitions" clause, enabling easy navigation to the term definition.

47

OCL usage conventions

48 Constraints in this document are specified using OCL (see OCL 2.0).

49 OCL statements are in `monospaced font`.

50

Enabled Logical Element Profile

51

1 Scope

52 The Enabled Logical Element Profile is a pattern profile that extends the management capabilities of referencing profiles by adding the capability to represent any enabled logical element. This profile describes common requirements for modeling the variety of enabled logical elements within managed systems including enabled state management, health state, and operational status.

53

2 Normative references

54 The following referenced documents are indispensable for the application of this document. For dated or versioned references, only the edition cited (including any corrigenda or DMTF update versions) applies. For references without a date or version, the latest published edition of the referenced document (including any corrigenda or DMTF update versions) applies.

55 DMTF DSP0004, *CIM Infrastructure Specification 2.7*,
http://www.dmtf.org/standards/published_documents/DSP0004_2.7.pdf

56 DMTF DSP0223, *Generic Operations 1.1*,
http://www.dmtf.org/standards/published_documents/DSP0223_1.1.pdf

57 DMTF DSP1001, *Management Profile Specification Usage Guide 1.2*,
http://www.dmtf.org/standards/published_documents/DSP1001_1.2.pdf

58 DMTF DSP1103, *Job Control Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1103_1.0.pdf

59 OMG formal/06-05-01, *Object Constraint Language 2.0*,
http://www.omg.org/spec/OCL/2.0/

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

61

3 Terms and definitions

62 In this document, some terms have a specific meaning beyond the normal English meaning. Those terms are defined in this clause.

63

3.1 General

64 The terms "shall" ("required"), "shall not", "should" ("recommended"), "should not" ("not recommended"), "may", "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described in ISO/IEC Directives, Part2, Annex H. The terms in parenthesis are alternatives for the preceding term, for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that ISO/IEC Directives, Part2, Annex H specifies additional alternatives. Occurrences of such additional alternatives shall be interpreted in their normal English meaning in this document.

65 The terms "clause", "subclause", "paragraph", "annex" in this document are to be interpreted as described in ISO/IEC Directives, Part2, Clause 5.

66

The terms "normative" and "informative" in this document are to be interpreted as described in ISO/IEC Directives, Part2, Clause 3. In this document, clauses, subclauses or annexes indicated with "(informative)" as well as notes and examples do not contain normative content.

The terms defined in DSP0004, DSP0223, and DSP1001 apply to this document.

The following additional terms are defined in this document.

3.2

enabled logical element

a logical element that has a concept of enabled state associated with it.

4 Symbols and abbreviated terms

The abbreviations defined in DSP0004, DSP0223, and DSP1001 apply to this document.

This document does not define any additional abbreviations.

5 Synopsis

Profile name: Enabled Logical Element

Version: 2.0.0

Organization: DMTF

Abstract: No

Profile type: Pattern

Schema: DMTF CIM 2.24

The Enabled Logical Element profile is a pattern profile that extends the management capability of the referencing profiles by adding a common representation of enabled logical elements.

Table 1 identifies the profile references defined in this profile.

Table 1 – Profile references

Profile reference name	Profile name	Organization	Version	Relationship	Description
JobControl	Job Control	DMTF	1.0	Conditional	Condition: The AsynchronousRequestStateChange feature is implemented.

Table 2 identifies the features defined in this profile.

Table 2 – Features

Feature	Requirement	Description
Capabilities	Conditional	See 7.1.1.
EnabledStateRepresentation	Conditional	See 7.1.2.
EnabledStateManagement	Conditional	See 7.1.3.
AsynchronousRequestStateChange	Optional	See 7.1.4.

Feature	Requirement	Description
ElementNameRepresentation	Conditional	See 7.1.5.
ElementNameModification	Optional	See 7.1.6.

87 Table 3 identifies the class adaptations defined in this profile.

88 **Table 3 – Adaptations**

Adaptation	Elements	Requirement	Description
Instantiated, embedded and abstract adaptations			
EnabledLogicalElement	CIM_EnabledLogicalElement	Mandatory	See 7.2.2.
ElementCapabilities	CIM_ElementCapabilities	Conditional	See 7.2.3.
EnabledLogicalElementCapabilities	CIM_EnabledLogicalElementCapabilities	Conditional	See 7.2.4.
ConcreteJob	CIM_ConcreteJob	See embedding elements	See 7.2.5.
Indications and exceptions			
This profile does not define any such adaptations.			

89 Table 4 identifies the use cases and state descriptions defined in this profile.

90 **Table 4 – Use cases and state descriptions**

Name	Description
State description: SimpleScenario	See 8.1.
State description: ResetStateTransitions	See 8.2.
Use case: DetermineLevelOfStateManagement	See 8.3.
Use case: EnableElement	See 8.4.
Use case: DisableElement	See 8.5.
Use case: ResetElement	See 8.6.
Use case: DetermineElementNameModifiable	See 8.7.

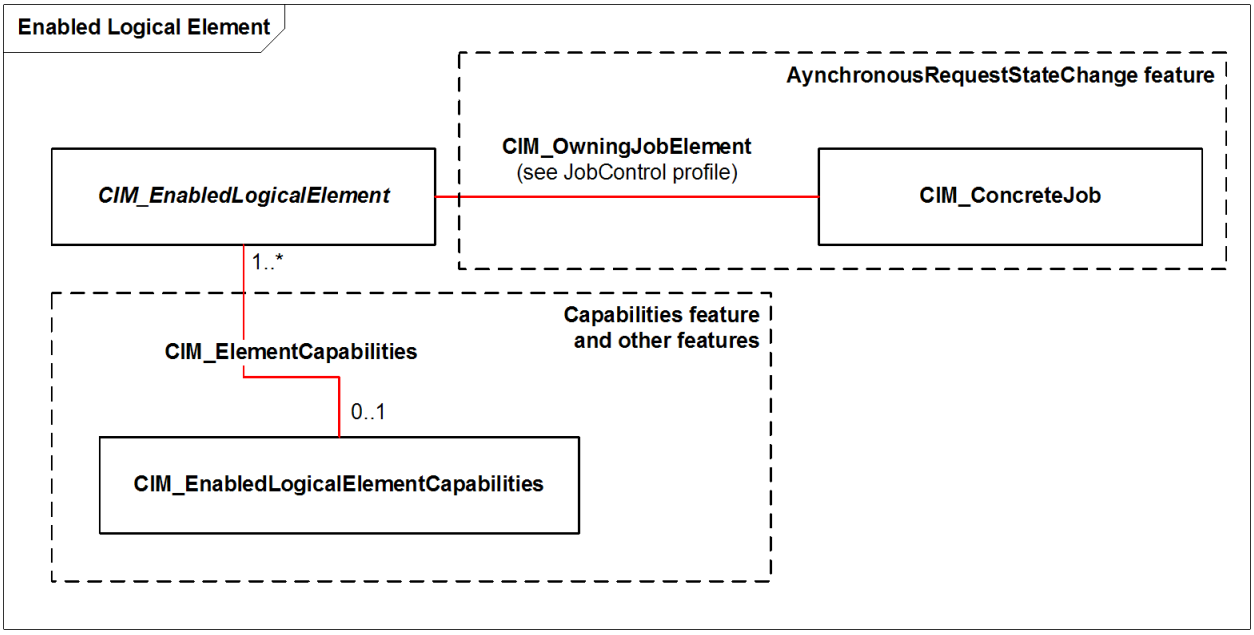
91

6 Description

92 The Enabled Logical Element profile is a pattern profile that describes the common set of attributes and behavior for enabled logical elements. The profile also specifies a set of properties representing the enabled state, the requested state and the current operational and health status of managed elements, an optional method for the initiation of enabled state changes, and an optional capability class conveying information about supported requested states and support for client state management and client modification of the ElementName property.

93 The adaptation diagram in Figure 1 shows all class usages (adaptations) defined in this profile.

94



95 **Figure 1 – Adaptation diagram**

96

The EnabledLogicalElement class adaptation contains properties to represent the enabled state, different aspects of the operational status, and health state. The EnabledLogicalElementCapabilities adaptation associated to the EnabledLogicalElement through ElementCapabilities represents the capabilities of the associated enabled logical element.

97

6.1 State and status properties

98 The current state and status of an enabled logical element is represented using the following properties on its EnabledLogicalElement instance:

99 • EnabledState, representing the enabled state of the element, using values such as Enabled and Disabled

100 • PrimaryStatus, representing the primary condition of the element, using values such as OK, Degraded, and Error that map to the commonly used "green", "yellow", and "red" conditions

101 • DetailedStatus, representing a more detailed condition of the element, that expands upon the value of the PrimaryStatus property

102 • HealthState, representing the health state of the element

103 • OperatingStatus, representing the precise operational status of the element and can be used for providing more detail with respect to the value of the EnabledState property

104 • CommunicationStatus, representing the ability of the implementation to communicate with the element.

105 • OperationalStatus (an array property), representing an operational status of the element. This property covers a number of different areas; the properties PrimaryStatus, DetailedStatus, OperatingStatus, and CommunicationStatus represent the same information in a more specific way and should be used in new profiles instead of OperationalStatus.

106

6.2 Consistency between PrimaryStatus, HealthState and OperationalStatus

107 Non-Null values of the properties PrimaryStatus, HealthState, and of the first array entry of OperationalStatus (that is, OperationalStatus[0]) need to be consistent as described in Table 5:

108 **Table 5 – Consistency between PrimaryStatus, HealthState and OperationalStatus[0]**

PrimaryStatus	HealthState	OperationalStatus[0]
0 (Unkown)	0 (Unkown)	0 (Unkown)
1 (OK)	5 (OK)	2 (OK)
2 (Degraded)	10 (Degraded/Warning)	3 (Degraded)
2 (Degraded)	15 (Minor Failure)	3 (Degraded)
3 (Error)	20 (Major Failure)	6 (Error)
3 (Error)	25 (Critical Failure)	6 (Error)
3 (Error)	30 (Non-recoverable Error)	6 (Error)

109

6.3 Consistency between CommunicationStatus and OperationalStatus

110 If the OperationalStatus property is non-Null (that is, implemented) and any of its array entries has one of the values listed in Table 6, the CommunicationStatus property if non-Null (that is, implemented) needs to have the corresponding value listed in Table 6.

111 **Table 6 – Consistency between CommunicationStatus and OperationalStatus[]**

OperationalStatus[] value	Required value for CommunicationStatus
12 (No Contact)	4 (No Contact)
13 (Lost Communication)	3 (Lost Communication)

112

7 Implementation

113

7.1 Features

114

7.1.1 Feature: Capabilities

115 **Requirement level:**

Conditional

116 **Condition:**

At least one of the following is true:

117 • The EnabledStateManagement feature is implemented.

118 • The ElementNameModification feature is implemented.

119 Implementing this feature for an enabled logical element provides support for representing the capabilities of the element.

120 This feature can be made available to clients at the granularity of EnabledLogicalElement instances.

121 It can be concluded that the feature is available for a EnabledLogicalElement instance if:

123 • The following OCL derivation constraint evaluates to a Boolean value of True.

124

OCL context: A EnabledLogicalElement instance.

```
derive: self.ElementCapabilities::Capabilities->size() > 0
```

Explanation:

An EnabledLogicalElementCapabilities instance exists that is associated with the EnabledLogicalElement instance through ElementCapabilities.

Otherwise, it can be concluded that the feature is not available.

7.1.2 Feature: EnabledStateRepresentation

Requirement level:

Conditional

Condition:

The EnabledStateManagement feature is implemented.

Implementing this feature for an enabled logical element provides support for representation of the enabled state of the element.

If this feature is implemented for an enabled logical element, the following constraints apply:

- The EnabledState property in EnabledLogicalElement shall not have the value 5 (Not Applicable), but a value that indicates the state of the element.
- The RequestedState property in EnabledLogicalElement shall not have the value 12 (Not Applicable), but a value that indicates the last requested state of the element.
- If the TransitioningToState property in EnabledLogicalElement is non-Null (that is, implemented), it shall not have the value 12 (Not Applicable), but a value that indicates the state transition of the element.

If this feature is not implemented for an enabled logical element, the following constraints apply:

- The EnabledState property in EnabledLogicalElement shall have the value 5 (Not Applicable).
- The RequestedState property in EnabledLogicalElement shall have the value 12 (Not Applicable).
- The TransitioningToState property in EnabledLogicalElement shall be Null or shall have the value 12 (Not Applicable).

This feature can be made available to clients at the granularity of EnabledLogicalElement instances.

It can be concluded that the feature is available for a EnabledLogicalElement instance if:

- The following OCL derivation constraint evaluates to a Boolean value of True.

  OCL context: A EnabledLogicalElement instance.

  ```
  derive: self.EnabledState != 5 /* Not Applicable */
  ```

  Explanation:

  The EnabledState property of the EnabledLogicalElement instance does not have the value 5 (Not Applicable).

Otherwise, it can be concluded that the feature is not available.

7.1.3 Feature: EnabledStateManagement

149 **Requirement level:**

Conditional

150 **Condition:**

The AsynchronousRequestStateChange feature is implemented.

151 Implementing this feature for an enabled logical element provides support for client management of the enabled state of the element.

152 Implementing this feature for an enabled logical element requires that the EnabledStateRepresentation and Capabilities features are also implemented for that element.

153 If this feature is implemented for an enabled logical element, the following constraints apply:

154 • If the EnabledLogicalElement. AvailableRequestedStates property is non-Null (that is, implemented), it shall contain zero or more of the values contained in the RequestedStatesSupported property of the associated EnabledLogicalElementCapabilities instance, where zero number of values indicates that there are no available requested states. It shall not contain any other values.

155 • The RequestedStatesSupported property in EnabledLogicalElementCapabilities shall contain at least one value. The set of values in the array shall represent the supported values for the RequestedState parameter of the RequestStateChange( ) method. That is, for each value there exist conditions under which an invocation of that method with that parameter set to the value returns 0 (Completed with No Error).

156 • The RequestStateChange( ) method in EnabledLogicalElement shall be implemented and shall not return 1 (Unsupported).

157 If this feature is not implemented for an enabled logical element, the following constraints apply:

158 • The AvailableRequestedStates property in EnabledLogicalElement shall be Null.

159 • If there is an instance of EnabledLogicalElementCapabilities associated with the EnabledLogicalElement instance, its RequestedStatesSupported property shall be Null.

160 • The RequestStateChange( ) method in EnabledLogicalElement shall either not be implemented or if implemented, shall return 1 (Unsupported).

161 This feature can be made available to clients at the granularity of EnabledLogicalElement instances.

162 It can be concluded that the feature is available for a EnabledLogicalElement instance if:

164 • The following OCL derivation constraint evaluates to a Boolean value of True.

165 OCL context: A EnabledLogicalElement instance.

166
```
derive: self.ElementCapabilities::Capabilities.RequestedStatesSupported->size()
> 0
```

167 Explanation:

An EnabledLogicalElementCapabilities instance exists that is associated with the EnabledLogicalElement instance through ElementCapabilities, and the RequestedStatesSupported property of that EnabledLogicalElementCapabilities instance contains at least one value.

168 Otherwise, it can be concluded that the feature is not available.

169

7.1.4 Feature: AsynchronousRequestStateChange

170 **Requirement level:**

Optional

171 Implementing this feature for an enabled logical element provides support for asynchronous execution of the RequestStateChange( ) method of the EnabledLogicalElement adaptation.

172 Implementing this feature for an enabled logical element requires that the EnabledStateManagement feature is also implemented for that element.

173 If this feature is implemented for an enabled logical element, the RequestStateChange( ) method shall support asynchronous execution. Note that the method implementation may decide between synchronous and asynchronous execution on every invocation individually.

174 If this feature is not implemented for an enabled logical element, the RequestStateChange( ) method shall not support asynchronous execution; any invocations of the method shall be performed synchronously.

175 This feature can be made available to clients at the granularity of EnabledLogicalElement instances.

176 Availability of this feature cannot be discovered by clients (other than trying the functionality provided by the feature).

177

7.1.5 Feature: ElementNameRepresentation

178 **Requirement level:**

Conditional

179 **Condition:**

The ElementNameModification feature is implemented.

180 Implementing this feature for an enabled logical element provides support for representing its element name (that is, the ElementName property of the EnabledLogicalElement instance representing the element).

181 If this feature is implemented for an enabled logical element, the ElementName property shall be non-Null.

182 If this feature is not implemented for an enabled logical element, the ElementName property shall be Null.

183 This feature can be made available to clients at the granularity of EnabledLogicalElement instances.

184 It can be concluded that the feature is available for a EnabledLogicalElement instance if:

186 - The following OCL derivation constraint evaluates to a Boolean value of True.

187 OCL context: A EnabledLogicalElement instance.

188
```
derive: self.ElementName != Null
```

189 Explanation:

The ElementName property of the EnabledLogicalElement instance is non-Null.

190 Otherwise, it can be concluded that the feature is not available.

191

7.1.6 Feature: ElementNameModification

192 **Requirement level:**

Optional

193 Implementing this feature for an enabled logical element provides support for client modification of its element name (that is, the ElementName property of the EnabledLogicalElement instance representing the element).

194 Implementing this feature for an enabled logical element requires that the Capabilities feature is also implemented for that element.

195 If this feature is implemented for an enabled logical element, the following constraints apply:

196 • The ElementNameEditSupported property in EnabledLogicalElementCapabilities shall have the value True.

197 • The MaxElementNameLen property in EnabledLogicalElementCapabilities shall be non-Null (that is, implemented).

198 • The ElementNameMask property in EnabledLogicalElementCapabilities shall have a value that is a regular expression defined using the syntax specified in Annex B of DSP1001.

199 If this feature is not implemented for an enabled logical element, the following constraints apply:

200 • If there is an instance of EnabledLogicalElementCapabilities associated with the EnabledLogicalElement instance, its ElementNameEditSupported property shall have the value False.

201 This feature can be made available to clients at the granularity of EnabledLogicalElement instances.

202 It can be concluded that the feature is available for a EnabledLogicalElement instance if:

204 • The following OCL derivation constraint evaluates to a Boolean value of True.

205 OCL context: A EnabledLogicalElement instance.

206
```
derive: self.ElementCapabilities::Capabilities.ElementNameEditSupported = True
```

207 Explanation:

An EnabledLogicalElementCapabilities instance exists that is associated with the EnabledLogicalElement instance through ElementCapabilities, and the ElementNameEditSupported property of that EnabledLogicalElementCapabilities instance has the value True.

208 Otherwise, it can be concluded that the feature is not available.

209

7.2 Adaptations

210

7.2.1 Conventions

211 This profile defines operation requirements based on DSP0223.

212 For adaptations of ordinary classes and of associations, the requirements for operations are defined in adaptation-specific subclauses of subclause 7.2.

213 For association traversal operation requirements that are specified only in the elements table of an adaptation (i.e., without operation-specific subclauses), the names of the association adaptations to be traversed are listed in the elements table.

214 The default initialization requirement level for property requirements is optional.

215 The default modification requirement level for property requirements is optional.

216 This profile repeats the effective values of certain Boolean qualifiers as part of property, method parameter, or method return value requirements. The following convention is established: If the name of a

qualifier is listed, its effective value is True; if the qualifier name is not listed, its effective value is False. The convention is applied in the following cases:

- In: indicates that the parameter is an input parameter
- Out: indicates that the parameter is an output parameter
- Key: indicates that the property is a key (that is, its value is part of the instance path)
- Required: indicates that the element value shall be non-Null
- Null OK: indicates explicitly that the element value may be Null for mandatory, conditional or conditional exclusive properties. This information is not specified as a qualifier in the schema but as an indicator in the profile.

7.2.2 Adaptation: EnabledLogicalElement: CIM_EnabledLogicalElement

7.2.2.1 General

Adaptation type: Ordinary class

Implementation type: Instantiated

A concrete subclass of the abstract schema class CIM_EnabledLogicalElement needs to be implemented.

Requirement level:

Mandatory
This adaptation models enabled logical elements; that is, logical elements that have a concept of enabled state associated with them.

Constraints:

OCL constraint in the context of a EnabledLogicalElement instance:

```
inv: if self.ElementCapabilities::Capabilities.RequestedStatesSupported->size()
> 0
then self.RequestedState != 12 /* Not Applicable */
    and self.EnabledState != 5 /* Not Applicable */
else
```

Table 7 – EnabledLogicalElement: Element requirements

Element	Requirement	Description
Base adaptations		
JobControl::CIM_ManagedElement (affected element)	Mandatory	See JobControl::CIM_ManagedElement (affected element).
JobControl::CIM_ManagedElement (owning element)	Mandatory	See JobControl::CIM_ManagedElement (owning element).
Properties		
ElementName	Conditional	See 7.2.2.2
EnabledState	Mandatory	See 7.2.2.3
RequestedState	Mandatory	See 7.2.2.4
TransitioningToState	Optional	See 7.2.2.5
AvailableRequestedStates	Optional	See 7.2.2.6
HealthState	Optional	See 7.2.2.7

Element	Requirement	Description
PrimaryStatus	Optional	See 7.2.2.8
DetailedStatus	Optional	
OperatingStatus	Optional	
CommunicationStatus	Optional	See 7.2.2.9
OperationalStatus	Optional	See 7.2.2.10
Methods		
RequestStateChange( )	Conditional	See 7.2.2.11
Operations		
GetInstance( )	Mandatory	
ModifyInstance( )	Conditional	See 7.2.2.12
EnumerateInstances( )	Mandatory	
EnumerateInstanceNames( )	Mandatory	
Associators( )	Conditional	See 7.2.2.13
AssociatorNames( )	Conditional	See 7.2.2.14
References( )	Conditional	See 7.2.2.15
ReferenceNames( )	Conditional	See 7.2.2.16

232 #### 7.2.2.2 Property: ElementName

233 **Requirement level:**

Conditional

234 **Condition:**

The ElementNameRepresentation feature is implemented.

235 The value of this property shall be formatted as a free-form string of variable length (pattern ".*").

236 The value of this property should be the name of the enabled logical element as it would be communicated to an end-user. The value of this property should contain an identifier that can be used by the end-user to differentiate that logical element from another logical element of the same type contained in or aggregated by the same system.

237 For example, if the logical element is a port on a computer system with 100 ports over subordinate systems (sub system 1 and sub system 2), then the ElementName property could have the value "port 43 on sub system 2". If the logical element were a processor on a blade system within a modular system with two processors per blade system, then the ElementName property could have the value "processor 2 on blade system 1".

238 #### 7.2.2.3 Property: EnabledState

239 **Requirement level:**

Mandatory

240 The value of this property indicates the current enabled state of the enabled logical element, using any of the values defined in its value map (see CIM Schema).

241 The description of the EnabledStateRepresentation feature defines additional requirements for this property.

242

Specializing profiles may constrain the allowable values for this property and may define particular interpretations of those values.

243 When the represented enabled logical element is in transition from one state to another, the enabled state of the element may be indeterminate. Thus, if the TransitioningToState property is non-Null, does not have the value 5 (No Change) or 12 (Not Applicable) which represents a state transition in progress, the EnabledState property shall have the value 0 (Unknown).

244

7.2.2.4 Property: RequestedState

245 **Requirement level:**

Mandatory

246 The value of this property indicates the last requested enabled state of the enabled logical element (as requested through the RequestStateChange( ) method), using any of the values defined in its value map (see CIM Schema).

247 If the implementation cannot represent the last requested enabled state, this property shall have the value 0 (Unknown).

248 The description of the EnabledStateRepresentation feature defines additional requirements for this property.

249 Specializing profiles may constrain the allowable values for this property and may define particular interpretations of those values.

250

7.2.2.5 Property: TransitioningToState

251 **Requirement level:**

Optional

252 The description of the EnabledStateRepresentation feature defines additional requirements for this property.

253

7.2.2.6 Property: AvailableRequestedStates

254 **Requirement level:**

Optional

255 If this property is non-Null (that is, implemented), the requirements for this property depend on whether the EnabledStateManagement feature is implemented for the represented enabled logical element.

256

7.2.2.7 Property: HealthState

257 **Requirement level:**

Optional
The value of this property indicates the health state of the represented enabled logical element, using any of the values defined in its value map (see CIM Schema).

258 The value of this property needs to be consistent with non-Null values of the first array entry of the OperationalStatus property, and with non-Null values of the PrimaryStatus property, as described in Table 5.

259

7.2.2.8 Property: PrimaryStatus

260 **Requirement level:**

Optional
The value of this property indicates the primary condition of the represented enabled logical element, using any of the values defined in its value map (see CIM Schema).

261 The value of this property needs to be consistent with non-Null values of the first array entry of the OperationalStatus property, and with non-Null values of the HealthState property, as described in Table 5.

262

7.2.2.9 Property: CommunicationStatus

263 **Requirement level:**

Optional
The value of this property indicates the ability of the implementation to communicate with the represented enabled logical element, using any of the values defined in its value map (see CIM Schema).

264 The value of this property needs to be consistent with communication related values in any array entry of the OperationalStatus property, as described in Table 6.

265

7.2.2.10 Property: OperationalStatus

266 **Requirement level:**

Optional
The value of this property indicates the operational status of the represented enabled logical element, using any of the values defined in its value map (see CIM Schema). This property may contain values in its array entries that cover a number of different areas; the properties PrimaryStatus, DetailedStatus, OperatingStatus, and CommunicationStatus represent the same information in a more specific way and should be used in new profiles instead of OperationalStatus.

267 The value of the first array entry of this property (that is, OperationalStatus[0]) needs to be consistent with non-Null values of the PrimaryStatus property, and with non-Null values of the HealthState property, as described in Table 5.

268 **Constraints:**

269 OCL constraint in the context of a EnabledLogicalElement instance:

270
```
inv: self.OperationalStatus[0] in
{
    0 /* Unknown */,
    2 /* OK */,
    3 /* Degraded */,
    6 /* Error */
}
```

271 Explanation:

272 The value of the first array entry of this property (that is, OperationalStatus[0]) shall be one of the values listed in Table 8:

273 **Table 8 – Allowable values for OperationalStatus[0]**

OperationalStatus[0]
0 (Unkown)
2 (OK)
3 (Degraded)
6 (Error)

274

7.2.2.11 Method: RequestStateChange( )

275 **Requirement level:**

Conditional

276 **Condition:**

The EnabledStateManagement feature is implemented.

277 The requirements for this method depend on whether the EnabledStateManagement feature is implemented for the represented enabled logical element.

278 Invoking this method multiple times could result in earlier requests being overwritten or lost.

279 This method may perform its task asynchronously as part of a job that is started, or synchronously as part of the method invocation. If a job is started, the Job parameter on return references a ConcreteJob instance representing the started job. The implementation may support both or only one of these two modes of operation.

280 **Table 9 – RequestStateChange( ): Parameter requirements**

Parameter	Description
RequestedState	In, see 7.2.2.11.1
Job	Out, see 7.2.2.11.2
TimeoutPeriod	In, see 7.2.2.11.3
return value	See 7.2.2.11.4

281

7.2.2.11.1 Parameter: RequestedState

282 Specializing profiles may constrain the allowable values for this parameter and may define particular interpretations of those values.

283

7.2.2.11.2 Parameter: Job

A (non-Null) instance path to the ConcreteJob instance representing the job, if a job was started. Null, if no job was started.

284 **Constraints:**

285 Referenced instances shall be of class adaptation ConcreteJob.

286

7.2.2.11.3 Parameter: TimeoutPeriod

287 Client-specified maximum amount of time the transition to a new state is supposed to take:

288 • Null or interval 0 – No maximum time is specified

289 • Non-0 interval – The value specifies the maximum time allowed

290

7.2.2.11.4 Return value

291 This method shall return one of the values specified in Table 10:

292 **Table 10 – EnabledLogicalElement.RequestStateChange( ): Return values**

Value	Description
0	The state change was successfully performed and no job was started.

Value	Description
1	The method is not implemented.
2	An error has occurred (requirements for using this value are defined after this table).
4096	A job was started.

293 This method shall return the value 2 (Unknown or Unspecified Error) in any of the following cases:

294 • if the RequestedState parameter is Null.

295 • if the RequestedState parameter has a value that is not contained in the RequestedStatesSupported array property of the associated EnabledLogicalElementCapabilities instance.

297 • if the RequestedState parameter has a non-Null value that is not contained in the AvailableRequestedStates array property.

298 #### 7.2.2.12 Operation: ModifyInstance( )

299 **Requirement level:**

Conditional

300 **Condition:**

The ElementNameModification feature is implemented.

301 If the ElementNameModification feature is implemented for the represented enabled logical element, the ElementName property shall be modifiable, and this operation shall enforce the length restriction specified in the MaxElementNameLen property and the name format specified in the ElementNameMask property of the associated EnabledLogicalElementCapabilities instance.

302 #### 7.2.2.13 Operation: Associators( )

303 **Requirement level:**

Conditional

304 **Condition:**

The Capabilities feature is implemented.

305 #### 7.2.2.14 Operation: AssociatorNames( )

306 **Requirement level:**

Conditional

307 **Condition:**

The Capabilities feature is implemented.

308 #### 7.2.2.15 Operation: References( )

309 **Requirement level:**

Conditional

310 **Condition:**

The Capabilities feature is implemented.

311

7.2.2.16 Operation: ReferenceNames( )

312 **Requirement level:**

Conditional

313 **Condition:**

The Capabilities feature is implemented.

314

7.2.3 Adaptation: ElementCapabilities: CIM_ElementCapabilities

315

7.2.3.1 General

316 **Adaptation type:** Association class

317 **Implementation type:** Instantiated

318 **Requirement level:**

Conditional

319 **Condition:**

The Capabilities feature is implemented.

This adaptation models the relationship between enabled logical elements represented by EnabledLogicalElement instances and their capabilities represented by EnabledLogicalElementCapabilities instances.

320 **Table 11 – ElementCapabilities: Element requirements**

Element	Requirement	Description
Properties		
ManagedElement	Mandatory	Key, see 7.2.3.2
Capabilities	Mandatory	Key, see 7.2.3.3
Operations		
GetInstance( )	Mandatory	
EnumerateInstances( )	Mandatory	
EnumerateInstanceNames( )	Mandatory	

321

7.2.3.2 Property: ManagedElement

322 **Requirement level:**

Mandatory

323 **Constraints:**

324 • Referenced instances shall be of class adaptation EnabledLogicalElement.

325 • The multiplicity of this association end is 1 .. *

326

7.2.3.3 Property: Capabilities

327 **Requirement level:**

Mandatory

328 **Constraints:**

- Referenced instances shall be of class adaptation EnabledLogicalElementCapabilities.
- The multiplicity of this association end is 0 .. 1

7.2.4 Adaptation: EnabledLogicalElementCapabilities: CIM_EnabledLogicalElementCapabilities

7.2.4.1 General

Adaptation type: Ordinary class

Implementation type: Instantiated

Requirement level:

Conditional

Condition:

The Capabilities feature is implemented.

This adaptation models the capabilities of enabled logical elements.

Table 12 – EnabledLogicalElementCapabilities: Element requirements

Element	Requirement	Description
Properties		
InstanceID	Mandatory	Key
RequestedStatesSupported	Optional	See 7.2.4.2
ElementNameEditSupported	Mandatory	See 7.2.4.3
MaxElementNameLen	Conditional	See 7.2.4.4
ElementNameMask	Conditional	See 7.2.4.5
Operations		
GetInstance( )	Mandatory	
EnumerateInstances( )	Mandatory	
EnumerateInstanceNames( )	Mandatory	
Associators( )	Mandatory	
AssociatorNames( )	Mandatory	
References( )	Mandatory	
ReferenceNames( )	Mandatory	

7.2.4.2 Property: RequestedStatesSupported

Requirement level:

Optional

If this property is non-Null (that is, implemented), the requirements for this property depend on whether the EnabledStateManagement feature is implemented for the represented enabled logical element.

Specializing profiles may constrain the allowable set of values for this property.

7.2.4.3 Property: ElementNameEditSupported

Requirement level:

Mandatory

345 #### 7.2.4.4 Property: MaxElementNameLen

346 **Requirement level:**

Conditional

347 **Condition:**

The ElementNameModification feature is implemented.

349 #### 7.2.4.5 Property: ElementNameMask

350 **Requirement level:**

Conditional

351 **Condition:**

The ElementNameModification feature is implemented.

353 ### 7.2.5 Adaptation: ConcreteJob: CIM_ConcreteJob

354 This adaptation models a job that performs the task of a method invocation asynchronously.

355 **Adaptation type:** Ordinary class

356 **Implementation type:** Embedded

357 **Requirement level:**

Defined by its embedding elements

358 **Table 13 – ConcreteJob: Element requirements**

Element	Requirement	Description
Base adaptations		
JobControl::CIM_ConcreteJob	Mandatory	See JobControl::CIM_ConcreteJob.

359

8 Use cases and state descriptions

360 ## 8.1 State description: SimpleScenario

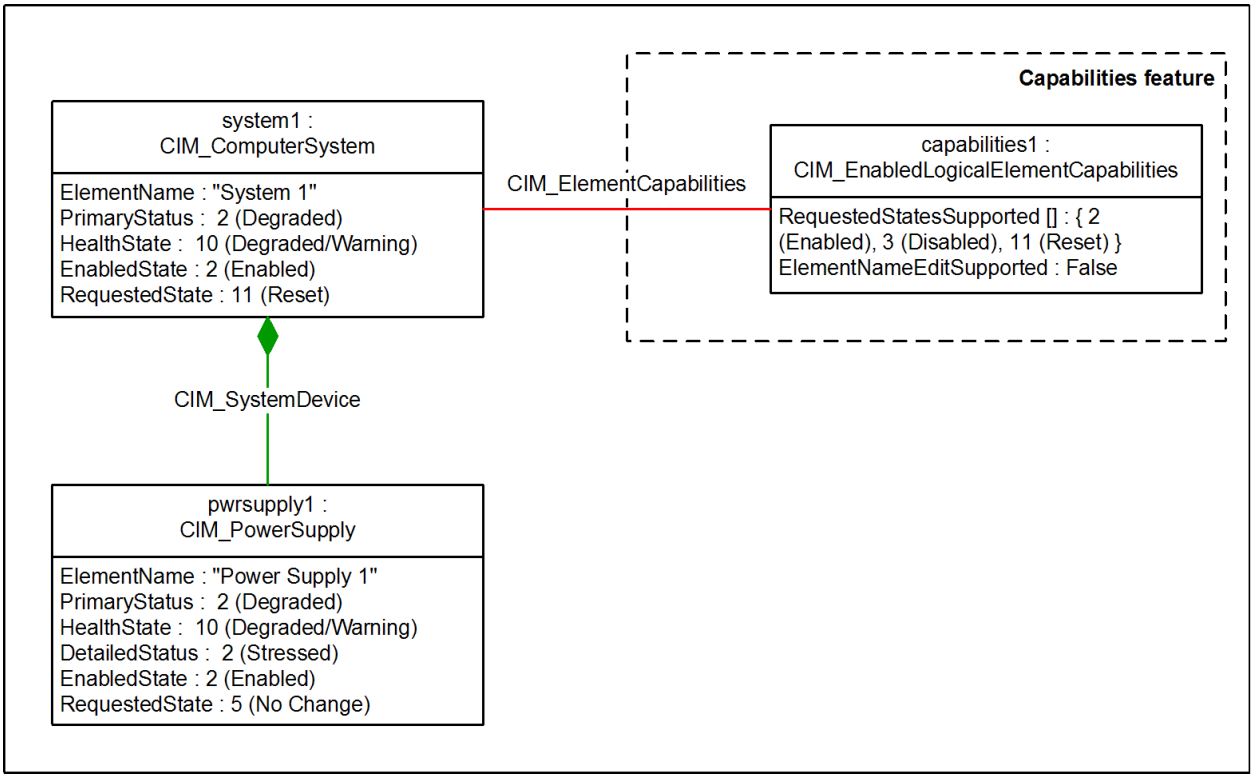
361 Figure 2 shows an object diagram for a simple scenario with enabled logical elements conforming to this profile.

362 The EnabledStateManagement feature has been implemented for `system1`, as indicated by the fact that values are present in the RequestedStatesSupported array property of `capabilities1`, and per these values the system represented by `system1` can be enabled, disabled and reset. The ElementNameModification feature has not been implemented for `system1` (as indicated by the value False of the ElementNameEditSupported property). The system represented by `system1` has been previously requested to be reset (the RequestedState property has a value of 11 (Reset)), but is currently still enabled (the EnabledState property has a value of 2 (Enabled)), with degraded status as indicated by its PrimaryStatus property, whose value is correctly derived from the value of the HealthState property.

363

The EnabledStateManagement feature has not been implemented for `pwrsupply1`. The power supply represented by `pwrsupply1` is also degraded, and reports a more granular status with the DetailedStatus property.

364



365 **Figure 2 – Object diagram for the SimpleScenario state description**

366

8.2 State description: ResetStateTransitions

367 This subclause describes possible state transitions during the reset of an enabled logical element that is initially in the enabled state.

368

8.2.1 Introduction and initial state

369 Figure 3 shows an object diagram with a system and a battery in its initial state. The `battery1` instance representing the battery is an enabled logical element and conforms to this profile. The diagram shows only properties that are relevant for the discussion of the state transitions, so some mandatory properties are not shown.

370 The `battery1` instance represents a battery; its EnabledState property has the value 2 (Enabled), indicating that the battery is currently enabled.

371 The RequestedState property has the value 0 (Unknown), indicating that the last requested state transition for `battery1` is unknown.

372 The AvailableRequestedStates array property contains the values of the enabled states the battery can transition to, given its its current enabled state. The RequestedStatesSupported property of the `capabilities1` instance advertises all the enabled states that are possible for `battery1`, regardless of its current enabled state.

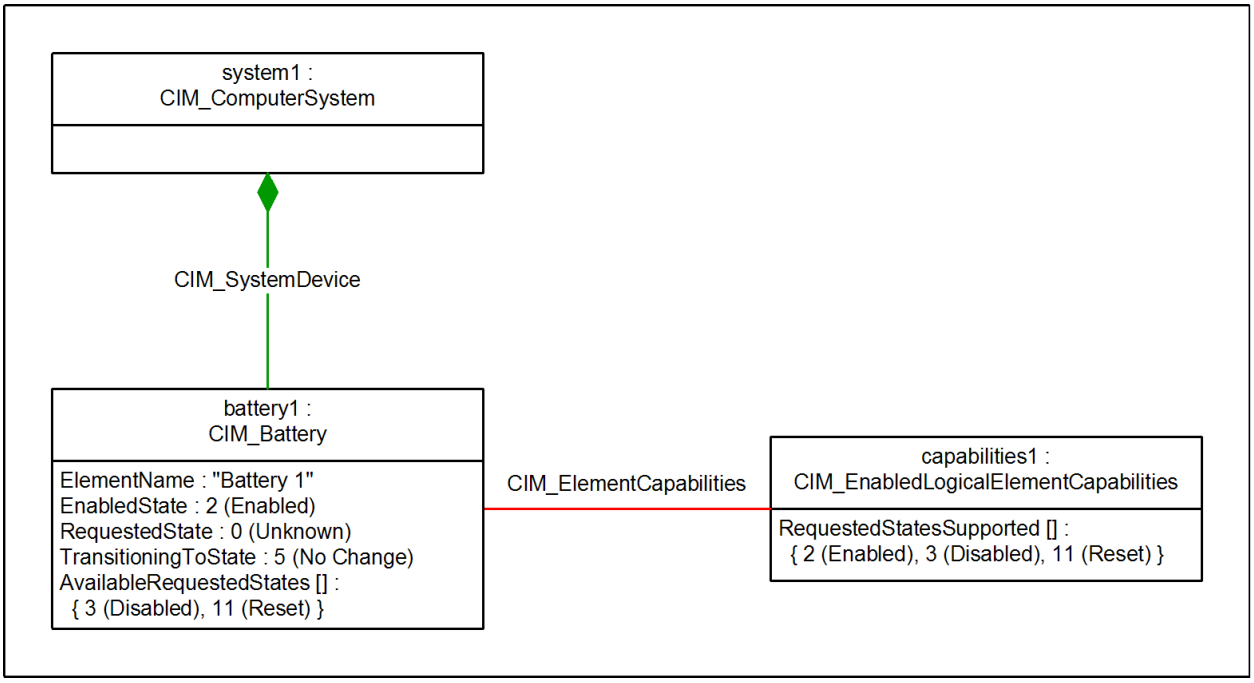
373

The battery represented by `battery1` is currently not in transition to any other enabled state, as indicated by the value 5 (No Change) of its TransitioningToState property. A state transition could be initiated by executing the RequestStateChange( ) method, because the EnabledStateManagement feature has been implemented for `battery1`. Note that a synchronous execution of the RequestStateChange( ) method requires having the state transition completed upon return of the method. As a result, the transitioning states (see 8.2.3 and 8.2.5) cannot be observed by the same client that invokes a synchronously executing RequestStateChange( ) method.

374 Subclauses 8.2.2, 8.2.3, 8.2.4, 8.2.5, and 8.2.6 describe the different states that `battery1` could go through after the successful execution of the RequestStateChange( ) method with the RequestedState parameter set to 11 (Reset), regardless whether the method execution is performed synchronously or asynchronously.

375 Note that the RequestedStatesSupported property of the `capabilities1` instance does not change regardless of the current enabled state of `battery1`, in contrast to the AvailableRequestedStates property of the `battery1` instance which changes depending on the current enabled state of `battery1`.

376



377 **Figure 3 – Object diagram for ResetStateTransitions: Original state**

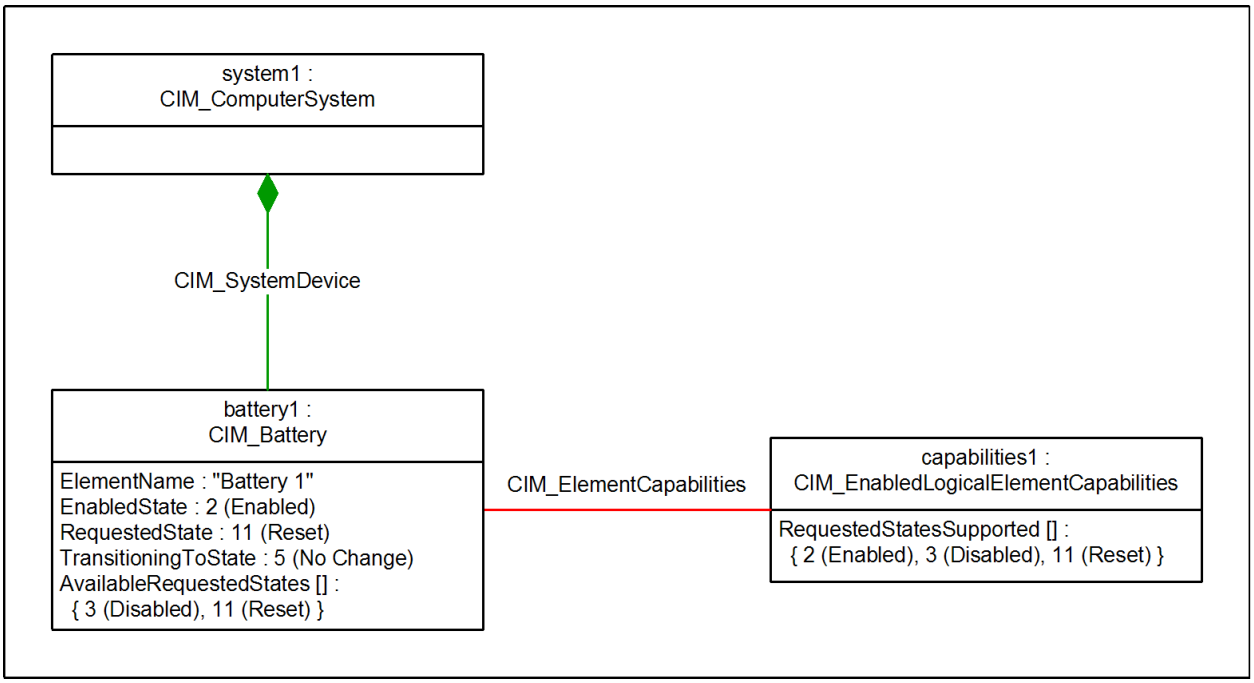
378

8.2.2 Successful reset request

379 Figure 4 shows an object diagram where `battery1` has successfully received the state transitioning request to 11 (Reset) as a result of the successful execution of the RequestStateChange( ) method with the RequestedState parameter set to 11 (Reset) as indicated by the value 11 (Reset) of the RequestedState property.

380 The EnabledState property of `battery1` still has a value of 2 (Enabled) and the TransitioningToState property still has a value of 5 (No Change), indicating that the battery is currently enabled and has not yet started the state transition.

381



382 **Figure 4 – Object diagram for ResetStateTransitions: After successful reset request**

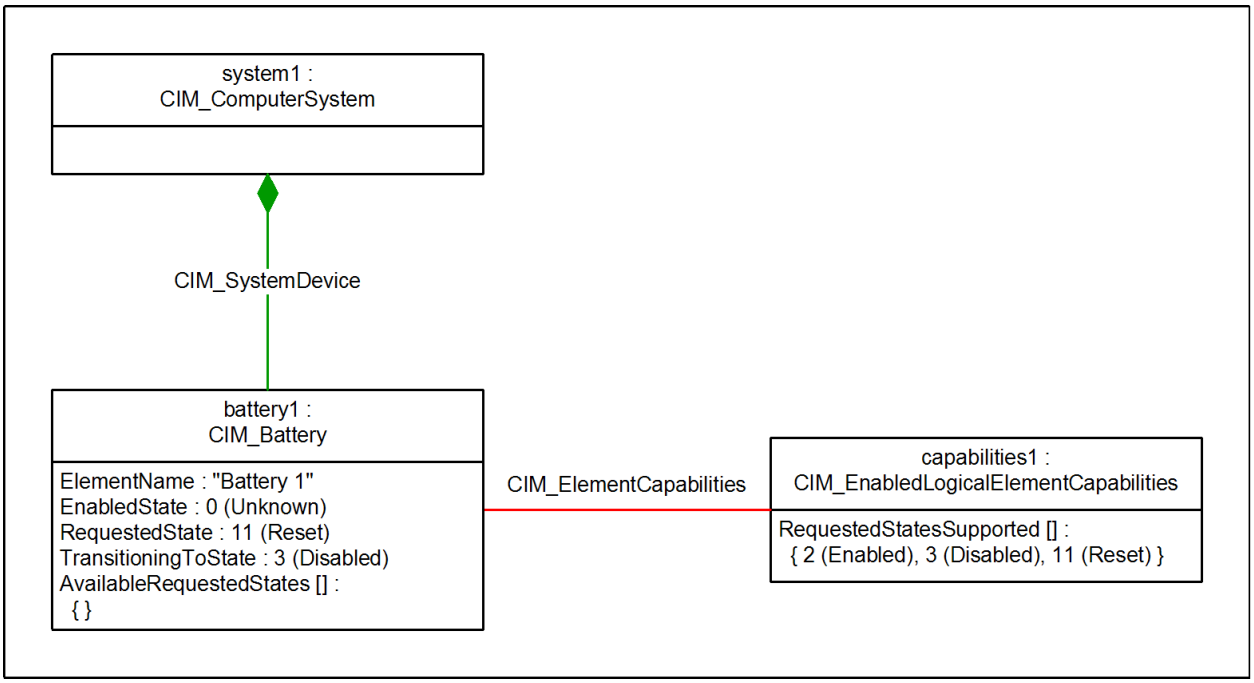
383

8.2.3 Transitioning to disabled state

384 Figure 5 shows an object diagram where `battery1` is meanwhile in transition to the disabled state.

385 The EnabledState property of `battery1` now has a value of 0 (Unknown) and the TransitioningToState property now has a value of 3 (Disabled), indicating that the battery is currently in transition to the disabled state. The AvailableRequestedStates property is an empty array indicating that the implementation does not accept any state change requests at this particular time.

386



387 **Figure 5 – Object diagram for ResetStateTransitions: Transitioning to disabled state**

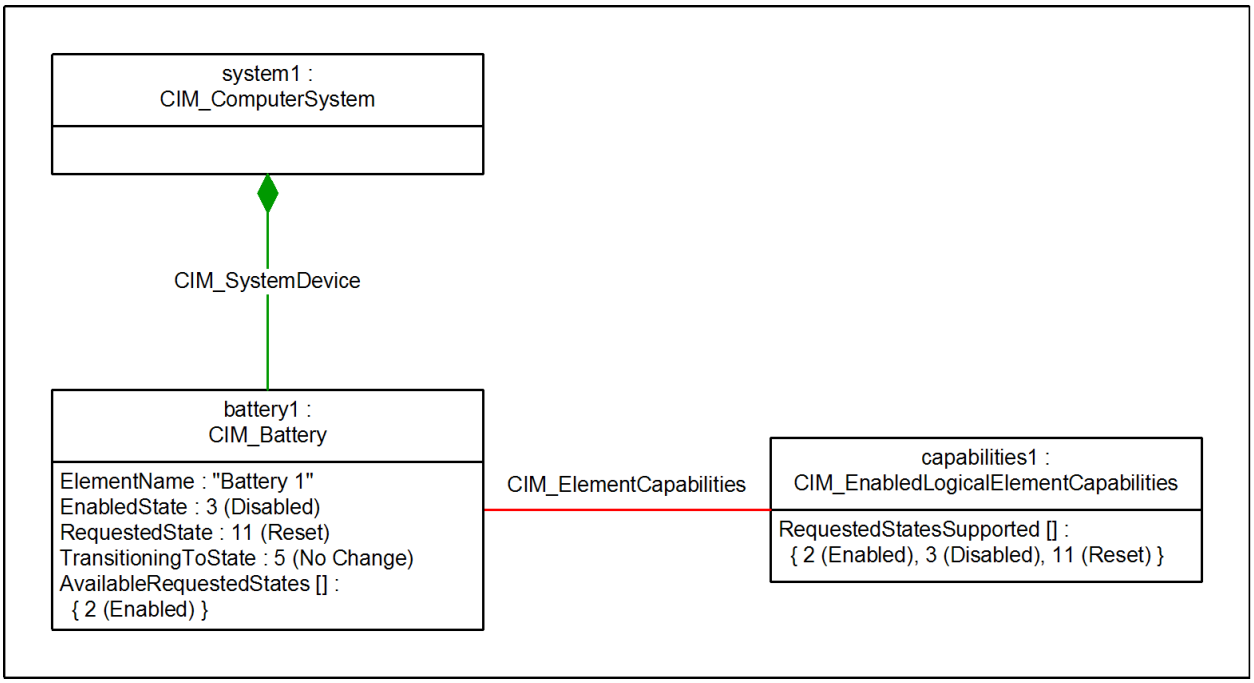
388

8.2.4 Transitioned to disabled state

389 Figure 6 shows an object diagram where `battery1` now has transitioned to the disabled state.

390 The EnabledState property of `battery1` now has a value of 3 (Disabled) and the TransitioningToState property now has a value of 5 (No Change), indicating that the battery is currently in the disabled state. The AvailableRequestedStates property contains the value 2 (Enabled), indicating that the implementation accepts state change requests to enable `battery1` at this particular time.

391



392 **Figure 6 – Object diagram for ResetStateTransitions: Transitioned to disabled state**

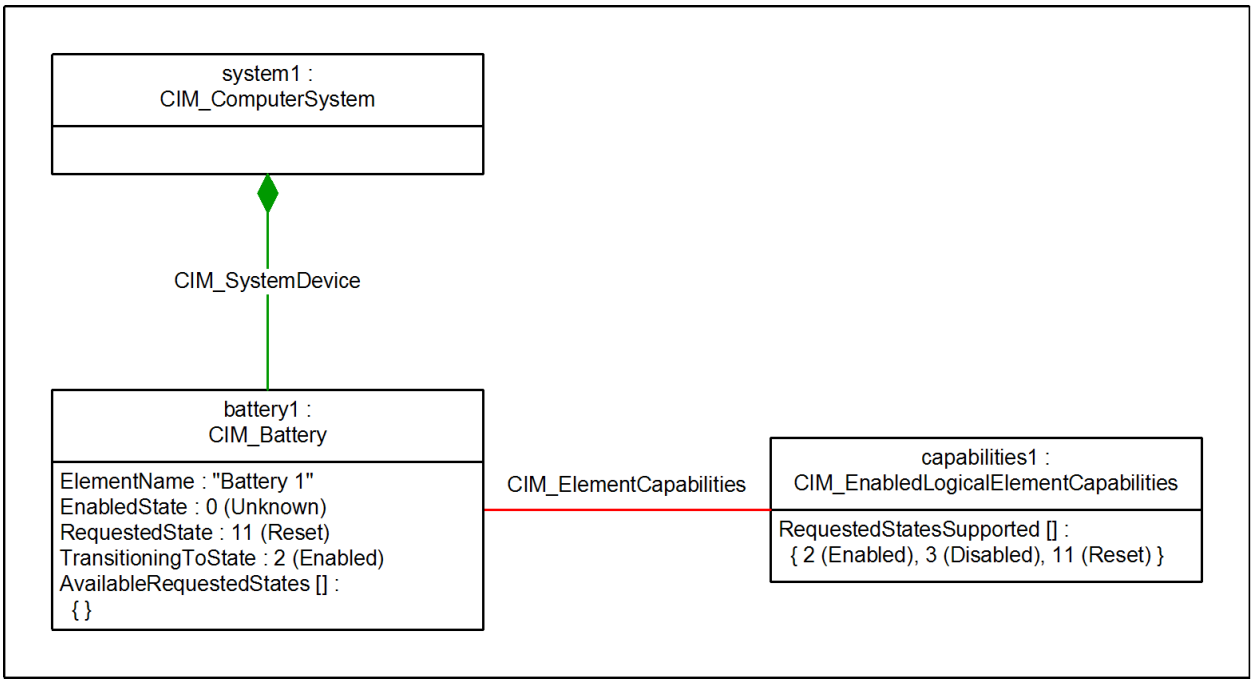
393

8.2.5 Transitioning to enabled state

394 Figure 7 shows an object diagram where `battery1` is now in transition back to the enabled state.

395 The EnabledState property of `battery1` now has a value of 0 (Unknown) and the TransitioningToState property now has a value of 2 (Enabled), indicating that `battery1` is currently in transition to the enabled state. The AvailableRequestedStates property is an empty array, indicating that the implementation does not accept any state change requests at this particular time.

396



397 **Figure 7 – Object diagram for ResetStateTransitions: Transitioning to enabled state**

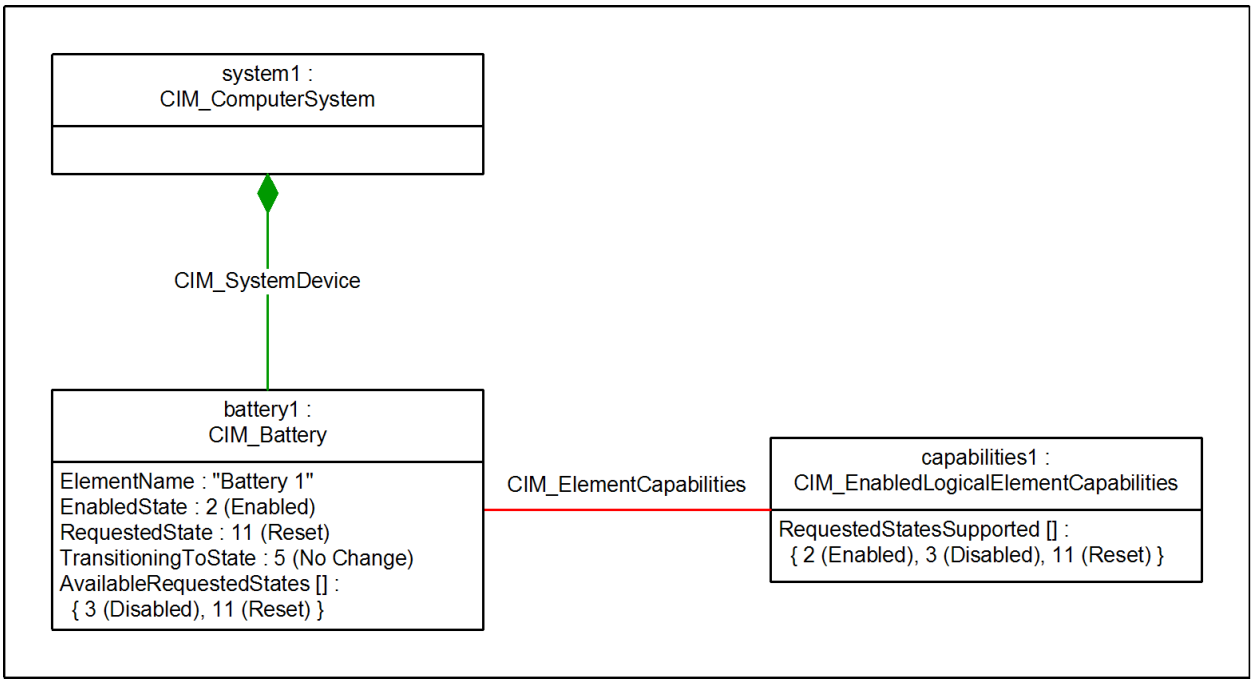
398

8.2.6 Transitioned to enabled state

399 Figure 8 shows an object diagram where `battery1` has arrived in its final state of the reset.

400 The EnabledState property of `battery1` now has a value of 2 (Enabled) again and the TransitioningToState property now has a value of 5 (No Change), indicating that `battery1` is currently in the enabled state. The AvailableRequestedStates property contains the values 3 (Disabled) and 11 (Reset), indicating that the implementation accepts disabling or resetting `battery1` at this particular time.

401



402 **Figure 8 – Object diagram for ResetStateTransitions: Transitioned to enabled state**

403

8.3 Use case: DetermineLevelOfStateManagement

This use case describes how a client can determine the level of state representation and state management that is supported for a particular enabled logical element.

404 This use case has the following preconditions:

405 • An instance of CIM_EnabledLogicalElement is known, representing the enabled logical element.

406 The main flow for this use case consists of the following steps:

407 1. For the given CIM_EnabledLogicalElement instance, retrieve the values of its EnabledState and RequestedState properties.

409 2. If the EnabledState and RequestedState properties do not have the value 12 (Not Applicable), the representation of enabled state (see the EnabledStateRepresentation feature) is supported for that enabled logical element; continue with step 3.

410 Otherwise, neither representation nor management of enabled state is supported for that enabled logical element, and the use case is complete.

411 3. Find the associated instance of CIM_EnabledLogicalElementCapabilities.

413 4. If the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported property is a non-empty array, management of enabled state (see the EnabledStateManagement feature) is supported for that enabled logical element (in addition to representation of enabled state).

414 Otherwise, management of enabled state is not supported for that enabled logical element.

415

8.4 Use case: EnableElement

This use case describes how a client can enable a particular enabled logical element.

416 This use case has the following preconditions:

- An instance of CIM_EnabledLogicalElement is known, representing the enabled logical element.

The main flow for this use case consists of the following steps:

1. Determine whether management of enabled state is supported for the enabled logical element, as described in the DetermineLevelOfStateManagement use case.
2. If management of enabled state is supported for the enabled logical element, continue to step 3.

   Otherwise, the state of the element cannot be changed by the client, and the use case ends.
3. If the AvailableRequestedStates property of the given CIM_EnabledLogicalElement instance contains the value 2 (Enabled), continue to step 4.

   Otherwise, the implementation supports enabling the enabled logical element but cannot transition to the enabled state at this time, and the use case ends.
4. Execute the RequestStateChange( ) method with the value of the RequestedState parameter set to 2 (Enabled), which requests to enable the enabled logical element.
5. If the RequestStateChange( ) method execution returns 0 (Success), the implementation has successfully processed the request to transition the enabled logical element's state to 2 (Enabled).

   If the RequestStateChange( ) method execution returns 4096 (Job Started), the implementation has started a job to perform the state transition asynchronously.

8.5 Use case: DisableElement

This use case describes how a client can disable a particular enabled logical element.

This use case has the following preconditions:

- An instance of CIM_EnabledLogicalElement is known, representing the enabled logical element.

The main flow for this use case consists of the following steps:

1. Determine whether management of enabled state is supported for the enabled logical element, as described in the DetermineLevelOfStateManagement use case.
2. If management of enabled state is supported for the enabled logical element, continue to step 3.

   Otherwise, the state of the element cannot be changed by the client, and the use case ends.
3. If the AvailableRequestedStates property of the given CIM_EnabledLogicalElement instance contains the value 3 (Disabled), continue to step 4.

   Otherwise, the implementation supports disabling the enabled logical element but cannot transition to the disabled state at this time, and the use case ends.
4. Execute the RequestStateChange( ) method with the value of the RequestedState parameter set to 3 (Disabled), which requests to disable the enabled logical element.
5. If the RequestStateChange( ) method execution returns 0 (Success), the implementation has successfully processed the request to transition the enabled logical element's state to 3 (Disabled).

   If the RequestStateChange( ) method execution returns 4096 (Job Started), the implementation has started a job to perform the state transition asynchronously.

8.6 Use case: ResetElement

This use case describes how a client can reset a particular enabled logical element.

This use case has the following preconditions:

- An instance of CIM_EnabledLogicalElement is known, representing the enabled logical element.

The main flow for this use case consists of the following steps:

1. Determine whether management of enabled state is supported for the enabled logical element, as described in the DetermineLevelOfStateManagement use case.
2. If management of enabled state is supported for the enabled logical element, continue to step 3.

   Otherwise, the state of the element cannot be changed by the client, and the use case ends.
3. If the AvailableRequestedStates property of the given CIM_EnabledLogicalElement instance contains the value 11 (Reset), continue to step 4.

   Otherwise, the implementation supports resetting the enabled logical element but cannot transition to the reset state at this time, and the use case ends.
4. Execute the RequestStateChange( ) method with the value of the RequestedState parameter set to 11 (Reset), which requests to reset the enabled logical element.
5. If the RequestStateChange( ) method execution returns 0 (Success), the implementation has successfully processed the request to transition the enabled logical element's state to 11 (Reset).

   If the RequestStateChange( ) method execution returns 4096 (Job Started), the implementation has started a job to perform the state transition asynchronously.

8.7 Use case: DetermineElementNameModifiable

This use case describes how a client can determine whether client modification of the ElementName property is supported for a particular enabled logical element.

This use case has the following preconditions:

- An instance of CIM_EnabledLogicalElement is known, representing the enabled logical element.

The main flow for this use case consists of the following steps:

1. Find the CIM_EnabledLogicalElementCapabilities instance that is associated with the given CIM_EnabledLogicalElement instance.
2. Examine the value of the ElementNameEditSupported property of the associated CIM_EnabledLogicalElementCapabilities instance.

   If the property value is True, the client can modify the value of the ElementName property of the given CIM_EnabledLogicalElement instance.

   Otherwise, the client cannot modify the value of the ElementName property of the given CIM_EnabledLogicalElement instance.

ANNEX A
(informative)

Change log

Table 14 – Change log

Version	Date	Description
1.0.0	2009-05-18	Published as DMTF Standard
2.0.0a	2014-01-14	Published as a Work in Progress, with the following changes: • Fixed the inconsistency that a central class was defined but DSP1033 was not referenced, by changing the profile type to the new profile type of "pattern profile": See DSP1001 1.2 for details on pattern profiles. • Fixed the issue that the ElementName property was Mandatory, by making it part of an optional feature "ElementNaming". • Fixed the issue that the PrimaryStatus and HealthState propertes were Mandatory, by making them Optional. • Added functionality to the Job output parameter of RequestStateChange() by tying it to the Job Control Profile. • Fixed the error in the description of method RequestStateChange() that the AvailableRequestedStates property was incorrectly attributed to class CIM_EnabledLogicalElementCapabilities. • Fixed the issue that the TransitioningToState property was left optional for the case that enabled state was not represented, by requiring that it is Null or 12 (Not Applicable) in this case. • Fixed errors in the diagrams and descriptions of the use cases and state descriptions. • Restricted the values of OperationalStatus[0] of EnabledLogicalElement to Unknown, OK, Degraded, and Error. • Established consistency requirements between the values of PrimaryStatus, HealthState and OperationalStatus[0] of EnabledLogicalElement. • Established consistency requirements between the values of CommunicationStatus and OperationalStatus[] of EnabledLogicalElement. • Converted to MRP XML format. • Using new generic operations names defined in DSP0223 1.0.2