

- IPMI -

IPMI CIM Mapping Guideline

Document Revision 0.60

2/6/2006

1 **Intellectual Property Agreement and Disclaimers**

2 The information contained in this IPMI CIM Mapping Guideline (“Mapping Guideline”) represents the current view
3 of Intel Corporation (“Intel”), Dell Incorporated (“Dell”), Avocent Corporation (“Avocent”) and Hewlett
4 Packard Company (“Hewlett Packard”) on issues discussed as of the date of publication. Because Intel, Dell,
5 Avocent and Hewlett Packard must respond to changing market conditions, it should not be interpreted to be a
6 commitment on the part of Intel, Dell, Avocent and Hewlett Packard, and Intel, Dell, Avocent and Hewlett Packard
7 cannot guarantee the accuracy of any information presented. This Mapping Guideline is for informational purposes
8 only.

9 Intel, Dell, Avocent and Hewlett Packard (collectively referred to herein as the “Authors”) provide this Mapping
10 Guideline under the following licensing terms. This Agreement to Royalty Free Licensing Terms for IPMI CIM
11 Mapping Guidelines Agreement and the corresponding Intellectual Property Agreement and Disclaimers for the
12 Mapping Guideline makes reference to the IPMI CIM Mapping Guideline. This Agreement to Royalty Free
13 Licensing Terms for IPMI CIM Mapping Guidelines Agreement and the corresponding Intellectual Property
14 Agreement and Disclaimers does not replace or supersede any of the Promoter, Contributor, or Adopter agreements,
15 Intellectual Property agreements, or disclaimers for any IPMI specifications, including, but not limited to, the
16 agreements for the Intelligent Platform Management Interface Specifications and the Intelligent Platform
17 Management Interface Second Generation Specifications.

18 THE MAPPING GUIDELINE IS PROVIDED “AS-IS” AND THE AUTHORS DO NOT MAKE ANY
19 REPRESENTATION OR WARRANTY REGARDING THE MAPPING GUIDELINE OR ANY PRODUCT OR
20 ITEM DEVELOPED BASED ON THE MAPPING GUIDELINE. THE AUTHORS DISCLAIM ALL EXPRESS
21 AND IMPLIED WARRANTIES, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF
22 MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND FREEDOM FROM
23 INFRINGEMENT. WITHOUT LIMITING THE GENERALITY OF THE FOREGOING, THE AUTHORS DO
24 NOT MAKE ANY WARRANTY OF ANY KIND THAT ANY ITEM DEVELOPED BASED ON THE MAPPING
25 GUIDELINE, OR ANY PORTION OF THEREOF, WILL NOT INFRINGE ANY COPYRIGHT, PATENT,
26 TRADE SECRET, OR OTHER INTELLECTUAL PROPERTY RIGHT OF ANY PERSON OR ENTITY IN ANY
27 COUNTRY EXCEPT AS PROVIDED FOR BELOW. IT IS YOUR RESPONSIBILITY TO SEEK LICENSES
28 FOR SUCH INTELLECTUAL PROPERTY RIGHTS WHERE APPROPRIATE.

29 THE AUTHORS SHALL NOT BE LIABLE FOR ANY DAMAGES ARISING OUT OF OR IN CONNECTION
30 WITH THE USE OF THE MAPPING GUIDELINE, INCLUDING LIABILITY FOR LOST PROFIT, BUSINESS
31 INTERRUPTION, OR ANY OTHER DAMAGES WHATSOEVER.

32 **Mapping Guideline Definitions.** “Necessary Claims” shall mean those claims of all patents and patent applications
33 throughout the world, excluding any claims to enabling technologies that may be necessary to make or use any
34 implementation of the Mapping Guideline but are not themselves expressly set forth in the Mapping Guideline (e.g.,
35 semiconductor manufacturing technology, basic operating system technology, etc.), which a party has the right to
36 grant licenses of the scope granted herein without such grant or the exercise of rights thereunder resulting in
37 payment of royalties or other consideration to third parties (except payment to affiliates) and which are necessarily
38 infringed by an implementation of this Mapping Guideline pursuant to the preceding and following sections of this
39 Intellectual Property Agreement, where such infringement could not have been avoided by another technically
40 reasonable non-infringing implementation of the Mapping Guideline.

41
42 **Grants of Licenses.** Each Author hereby grants, to each of the other Authors and all implementers of the Mapping
43 Guideline, a nonexclusive, perpetual, royalty-free, nontransferable, nonsublicenseable (except as part of transfer of
44 an end user product), worldwide license under its Necessary Claims to make, have made, use, import, offer to sell
45 and sell and otherwise distribute the portions of products, whether hardware, software, or combination of hardware
46 and software, which implement all or part of the Mapping Guideline for the sole purpose of mapping the IPMI 1.5
47 Specification into a DMTF CIM specification. No license is granted hereunder to any person or entity to implement
48 either IPMI or DMTF specifications.

49 Authors' grant of the above license to each implementer is subject to the implementer's agreement to grant, upon
50 request, a nonexclusive, perpetual, royalty-free, nontransferable, nonsublicenseable (except as part of transfer of an
51 end user product), worldwide license to the Authors, under such implementer's Necessary Claims to make, have
52 made, use, import, offer to sell and sell and otherwise distribute the portions of products, whether hardware,
53 software, or combination of hardware and software, which implement all or part of the same version of the
54 Mapping Guideline for the sole purpose of mapping the IPMI 1.5 Specification into a DMTF CIM specification.

55 **Document Feedback and Contributions.** Any feedback or other contribution by any person or entity in any
56 country that may be incorporated into this document can only be provided if the contributor agrees in writing to the
57 IPMI CIM Mapping Feedback Agreement. The IPMI CIM Mapping Feedback Agreement can be obtained by
58 contacting one of the Authors.

59 **Representation and warranties.** Each Author represents and warrants that to the actual knowledge of the Author's
60 named representative (1) the Author has full rights to make the contributions made by the Author to the Mapping
61 Guideline, (2) no other party has provided feedback or contributed to the Mapping Guideline through the Author and
62 that any party doing so in the future must have agreed to do so as a Contributing Party by executing the IPMI CIM
63 Mapping Feedback, and (3) the exercise of the licenses granted hereunder by the Author with regard to the Author's
64 contributions will not, to the knowledge of the Author, violate any third party copyright or trade secrets.

65 **Copyright.** The Authors hereby grant a nonexclusive, perpetual, irrevocable, non-transferable, worldwide royalty
66 free copyright license, to make copies, distribute, publish and display the Mapping Guideline so long as the below
67 copyright notice is retained with each copy of the Mapping Guideline.
68

69 © 2004–2005 Intel Corporation, Dell Incorporated, Avocent Corporation and Hewlett Packard Company. All rights
70 reserved.

71 This document is not for sale.

72
73 To obtain additional copies of this this document, please download the source files from the web sites at
74 <http://www.intel.com/design/servers/ipmi>

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. Members and non-members may reproduce DMTF specifications and documents for uses consistent with this purpose, provided that correct attribution is given. As DMTF specifications may be revised from time to time, the particular version and release date should always be noted.

Implementation of certain elements of DMTF standards or proposed standards may be subject to third party patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose, or identify any or all such third party patent right, owners or claimants, nor for any incomplete or inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize, disclose, or identify any such third party patent rights, or for such party's reliance on the standard or incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any party implementing such standard, whether such implementation is foreseeable or not, nor to any patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a patent owner for such implementations.

For information about patents held by third-parties which have notified the DMTF that, in their opinion, such patent may relate to or impact implementations of DMTF standards, visit <http://www.dmtf.org/about/policies/disclosures.php>.

Abstract

76 The Common Information Model (CIM)¹ is an object-oriented information model defined by the
77 Distributed Management Task Force (DMTF), which provides a conceptual framework for
78 describing management data.

79 This document defines CIM mappings for specific areas of server management. This includes
80 the behavioral conventions required for interoperability. This is intended to be a consistent,
81 cross-vendor mapping of the Intelligent Platform Management Interface (IPMI)² to CIM.

82 Referenced Standards

83 Cross-references to standards being published or developed by DMTF (or other organizations),
84 or reproduction of information from published standards, are for the purpose of interoperability
85 with those standards and for defining functionality needed for interoperability. Such cross-
86 references are not for the purpose of using copyrighted material from those standards. All such
87 referenced content remains outside of the scope of this document. See Reference Documents for
88 additional information.

89 Change History

Version	Date	Editor	Description
0.60	February 06, 2006	Jon R. Hass, Dell Inc.	Added the following major sections: <ul style="list-style-type: none">- IPMI Entity Mapping- IPMI Watchdog Mapping- IPMI Software Identity Mapping Updated Section 2 – Common Steps for Mapping Process. Changes to Sensor property mappings. Changes to Log property mappings. Changes to User ID Management mappings. Changes to BMC mappings. Changes to Versioning mapping behavior.

90 Open Issues

91

¹ Common Information Model (CIM) Specification, V2.2, June 14, 1999,
http://www.dmtf.org/standards/standard_cim.php

² Intelligent Platform Management Interface Specification, V1.5, February 20, 2002,
<http://www.intel.com/design/servers/ipmi>

92

Outline Table of Contents

93	Document Revision 0.60 2/6/2006	i
94	Intellectual Property Agreement and Disclaimers	ii
95	Abstract.....	iv
96	1 Introduction.....	1
97	2 Common Steps for Mapping Process.....	7
98	3 IPMI Sensor Mapping.....	20
99	4 IPMI Log Mapping.....	40
100	5 IPMI Commands Mapping.....	52
101	6 IPMI Power Control Mapping.....	55
102	7 IPMI User ID Management Mapping.....	61
103	8 IPMI BMC Mapping.....	69
104	9 IPMI Versioning Mapping.....	79
105	10 IPMI Entity Mapping.....	83
106	11 IPMI Watchdog Mapping.....	126
107	12 IPMI Software Identity Mapping.....	136
108	13 Acknowledgments.....	140
109		

110

Table of Contents

111	Abstract.....	iv
112	Change History.....	iv
113	Open Issues.....	iv
114	1 Introduction.....	1
115	1.1 Target Audience.....	1
116	1.2 Conventions.....	1
117	1.3 Reference Documents.....	1
118	1.4 Terminology.....	2
119	1.5 Acronyms and Abbreviations.....	2
120	1.6 Editorial Conventions.....	3
121	1.7 Notational Conventions.....	4
122	1.8 Referenced Standards.....	5
123	2 Common Steps for Mapping Process.....	7
124	2.1 Discovering IPMI Interfaces.....	7
125	2.1.1 System Interfaces.....	7
126	2.1.2 Remote Interfaces.....	7
127	2.2 Internal Interfaces.....	9
128	2.3 Discovery and Enumeration of IPMI Entity, Sensor, and FRU Information and	
129	Associations.....	9
130	2.3.1 Recommendations on the access and use of SDR Info.....	9
131	2.4 Entity Discovery.....	10

132	2.4.1	Entity presence rules.....	10
133	2.4.1.1	Explicit by Entity Association.....	11
134	2.4.1.2	Explicit by Sensor SDR or Event SDR.....	11
135	2.4.1.3	Explicit by FRU Device Locator Record.....	11
136	2.4.1.4	Implicit by Sensor Type.....	12
137	2.4.1.5	Implicit by FRU Data.....	12
138	2.5	Sensor Discovery.....	13
139	2.5.1	Sensor Presence Rules.....	13
140	2.6	FRU Device Discovery.....	14
141	2.6.1	By BMC Primary FRU Device.....	14
142	2.6.2	By Management Controller Device Locator Record.....	14
143	2.6.3	By FRU Device Locator Record.....	15
144	2.7	Discovering Sensor, Entity, & FRU Associations.....	15
145	2.8	Instantiating CIM Objects.....	17
146	2.8.1	Sensor Instantiation :.....	17
147	2.8.1.1	Sensor state/reading rules.....	18
148	2.8.2	Entity Instantiation.....	18
149	2.8.3	FRU instantiation.....	18
150	3	IPMI Sensor Mapping.....	20
151	3.1	Introduction.....	20
152	3.2	Instance Examples.....	21
153	3.3	Mapping Requirements.....	21
154	3.3.1	Handling Sensor Specific Offsets.....	21
155	3.4	CIM_Sensor Mapping.....	22
156	3.4.1	CIM_Sensor Methods.....	22
157	3.4.1.1	Method: Reset().....	22
158	3.4.1.2	Method: RequestStateChange.....	23
159	3.4.2	CIM_Sensor Properties.....	23
160	3.4.2.1	CIM_Sensor.SystemCreationClassName.....	24
161	3.4.2.2	CIM_Sensor.SystemName.....	24
162	3.4.2.3	CIM_Sensor.CreationClassName.....	25
163	3.4.2.4	CIM_Sensor.DeviceID.....	25
164	3.4.2.5	CIM_Sensor.ElementName and CIM_Sensor.Name.....	26
165	3.4.2.6	CIM_Sensor.SensorType and CIM_Sensor.OtherSensorTypeDescription.....	27
166	3.4.2.7	CIM_Sensor.PossibleStates.....	32
167	3.4.2.8	CIM_Sensor.CurrentState.....	33
168	3.4.2.9	CIM_Sensor.Caption.....	33
169	3.4.2.10	CIM_Sensor.Description.....	34
170	3.4.2.11	CIM_Sensor.PollingInterval.....	34
171	3.4.2.12	CIM_Sensor.HealthState.....	34
172	3.4.2.13	CIM_Sensor.OperationalStatus.....	34
173	3.5	CIM_NumericSensor Mapping.....	34
174	3.5.1	CIM_NumericSensor.Properties.....	35
175	3.5.1.1	NumericSensor.CurrentReading.....	36
176	3.5.1.2	NumericSensor.UnitModifier, BaseUnits, And RateUnits.....	36
177	3.5.1.3	NumericSensor.NominalReading.....	37
178	3.5.1.4	NumericSensor.NormalMax.....	37
179	3.5.1.5	NumericSensor.NormalMin.....	37
180	3.5.1.6	NumericSensor.MaxReadable.....	37
181	3.5.1.7	NumericSensor.MinReadable.....	37
182	3.5.1.8	NumericSensor.Resolution.....	37
183	3.5.1.9	NumericSensor.Accuracy.....	38

184	3.5.1.10	NumericSensor.IsLinear	38
185	3.5.1.11	NumericSensor.Hysteresis	38
186	3.5.1.12	NumericSensor.SupportedThresholds	38
187	3.5.1.13	NumericSensor.EnableThresholds	38
188	3.5.1.14	NumericSensor.SettableThresholds	38
189	3.5.1.15	NumericSensor.LowerThresholdNonCritical	38
190	3.5.1.16	NumericSensor.UpperThresholdNonCritical	39
191	3.5.1.17	NumericSensor.LowerThresholdCritical	39
192	3.5.1.18	NumericSensor.UpperThresholdCritical	39
193	3.5.1.19	NumericSensor.LowerThresholdFatal	39
194	3.5.1.20	NumericSensor.UpperThresholdFatal	39
195	4	IPMI Log Mapping	40
196	4.1	Introduction	40
197	4.2	Instance Diagrams	41
198	4.3	Mapping Requirements	41
199	4.3.1	Associating SEL with SDR	41
200	4.4	CIM_RecordLog Mapping	42
201	4.4.1	CIM_RecordLog.Methods	42
202	4.4.1.1	Method: ClearLog	42
203	4.4.1.2	Method: RequestStateChange	43
204	4.4.2	CIM_RecordLog.Properties	43
205	4.4.2.1	CIM_RecordLog.InstanceID	44
206	4.4.2.2	CIM_RecordLog.Name	44
207	4.4.2.3	CIM_RecordLog.MaxNumberOfRecords	44
208	4.4.2.4	CIM_RecordLog.CurrentNumberOfRecords	45
209	4.4.2.5	CIM_RecordLog.EnabledState	45
210	4.4.2.6	CIM_RecordLog.HealthState	45
211	4.4.2.7	CIM_RecordLog.OperationalStatus	45
212	4.4.2.8	CIM_RecordLog.Caption	45
213	4.4.2.9	CIM_RecordLog.Description	45
214	4.4.2.10	CIM_RecordLog.ElementName	45
215	4.5	CIM_LogRecord Mapping	45
216	4.5.1	CIM_LogRecord Methods	45
217	4.5.2	CIM_LogRecord Properties	46
218	4.5.2.1	CIM_LogRecord.LogCreationClassName	46
219	4.5.2.2	CIM_LogRecord.LogName	46
220	4.5.2.3	CIM_LogRecord.CreationClassName	46
221	4.5.2.4	CIM_LogRecord.RecordID	47
222	4.5.2.5	CIM_LogRecord.MessageTimestamp	47
223	4.5.2.6	CIM_LogRecord.RecordFormat	47
224	4.5.2.7	CIM_LogRecord.RecordData	48
225	4.5.2.8	CIM_LogRecord.Caption	48
226	4.5.2.9	CIM_LogRecord.Description	51
227	5	IPMI Commands Mapping	52
228	5.1	Introduction	52
229	5.2	Mapping Requirements	52
230	5.2.1	IPMI Commands Limitations	52
231	5.3	CIM_CommandService Mapping	52
232	5.3.1	CIM_CommandService.Methods	52
233	5.3.1.1	Method: ExecuteOEMCommand	52
234	5.3.2	CIM_CommandService.Properties	53

235	5.3.2.1	CIM_CommandService.SystemCreationClassName.....	54
236	5.3.2.2	CIM_CommandService.SystemName.....	54
237	5.3.2.3	CIM_CommandService.CreationClassName.....	54
238	5.3.2.4	CIM_CommandService.Name.....	54
239	5.3.2.5	CIM_CommandService.IANA.....	54
240	5.3.2.6	CIM_CommandService.Caption.....	54
241	5.3.2.7	CIM_CommandService.Description.....	54
242	6	IPMI Power Control Mapping.....	55
243	6.1	Introduction.....	55
244	6.2	Instance Diagrams.....	55
245	6.3	Mapping Requirements.....	56
246	6.4	CIM_PowerManagementService Mapping.....	56
247	6.4.1	CIM_PowerManagementService Methods.....	56
248	6.4.1.1	Method: PowerManagementService.SetPowerState().....	56
249	6.4.1.2	Method: PowerManagementService.StartService().....	57
250	6.4.1.3	Method: PowerManagementService.StopService().....	57
251	6.4.1.4	Method: PowerManagementService.RequestStateChange().....	57
252	6.4.2	CIM_PowerManagementService.Properties.....	57
253	6.4.2.1	CIM_PowerManagementService.CreationClassName.....	58
254	6.4.2.2	CIM_PowerManagementService.Name.....	58
255	6.5	CIM_PowerManagementCapabilities Mapping.....	58
256	6.5.1	CIM_PowerManagementCapabilities Methods.....	58
257	6.5.2	CIM_PowerManagementCapabilities Properties.....	58
258	6.5.2.1	CIM_PowerManagementCapabilities.PowerCapabilities.....	59
259	6.6	CIM_AssociatedPowerManagementService Mapping.....	59
260	6.6.1	CIM_AssociatedPowerManagementService Methods.....	59
261	6.6.2	CIM_AssociatedPowerManagementService Properties.....	59
262	6.6.2.1	CIM_AssociatedPowerManagementService.PowerState.....	60
263	6.6.2.2	CIM_AssociatedPowerManagementService.OtherPowerState.....	60
264	6.6.2.3	CIM_AssociatedPowerManagementService.PowerOnTime.....	60
265	7	IPMI User ID Management Mapping.....	61
266	7.1	Introduction.....	61
267	7.2	Instance Diagrams.....	61
268	7.3	Mapping Requirements.....	62
269	7.3.1	Mapping IPMI User ID's.....	62
270	7.3.2	Mapping IPMI Access Privileges.....	62
271	7.3.3	Modeling CIM_AuthorizedPrivilege.....	63
272	7.4	CIM_Account Mapping.....	63
273	7.4.1	CIM_Account Methods.....	63
274	7.4.2	CIM_Account Properties.....	63
275	7.4.2.1	CIM_Account.SystemCreationClassName.....	64
276	7.4.2.2	CIM_Account.SystemName.....	64
277	7.4.2.3	CIM_Account.CreationClassName.....	65
278	7.4.2.4	CIM_Account.Name.....	65
279	7.4.2.5	CIM_Account.UserID.....	65
280	7.4.2.6	CIM_Account.UserPassword.....	65
281	7.4.2.7	CIM_Account.OperationalStatus.....	65
282	7.4.2.8	CIM_Account.StatusDescriptions.....	66
283	7.5	CIM_Group Mapping.....	66
284	7.5.1	CIM_Group Methods.....	66
285	7.5.2	CIM_Group Properties.....	66

286	7.5.2.1	CIM_Group.CreationClassName.....	66
287	7.5.2.2	CIM_Group.Name.....	66
288	7.6	CIM_AuthorizedPrivilege Mapping.....	67
289	7.6.1	CIM_AuthorizedPrivilege Methods.....	67
290	7.6.2	CIM_AuthorizedPrivilege Properties.....	67
291	7.6.2.1	CIM_AuthorizedPrivilege.InstanceID.....	67
292	7.6.2.2	CIM_AuthorizedPrivilege.PrivilegeGranted.....	68
293	7.6.2.3	CIM_AuthorizedPrivilege.Activities.....	68
294	7.6.2.4	CIM_AuthorizedPrivilege.ElementName.....	68
295	8	IPMI BMC Mapping.....	69
296	8.1	Introduction.....	69
297	8.2	Instance Diagrams.....	69
298	8.3	Mapping Requirements.....	70
299	8.4	CIM_AdminDomain Mapping.....	71
300	8.4.1	CIM_AdminDomain Methods.....	71
301	8.4.1.1	Method: AdminDomain.RequestStateChange().....	71
302	8.4.2	CIM_AdminDomain.Properties.....	71
303	8.4.2.1	CIM_AdminDomain.CreationClassName.....	72
304	8.4.2.2	CIM_AdminDomain.Name.....	72
305	8.5	CIM_ComputerSystem Mapping.....	72
306	8.5.1	CIM_ComputerSystem Methods.....	72
307	8.5.1.1	Method: ComputerSystem.RequestStateChange().....	72
308	8.5.2	CIM_ComputerSystem Properties.....	73
309	8.5.2.1	CIM_ComputerSystem.Name.....	74
310	8.5.2.2	CIM_ComputerSystem.CreationClassName.....	75
311	8.5.2.3	CIM_ComputerSystem.Dedicated.....	75
312	8.5.2.4	CIM_ComputerSystem.Roles.....	75
313	8.5.2.5	CIM_ComputerSystem.EnabledState.....	76
314	8.5.2.6	CIM_ComputerSystem.HealthState.....	76
315	8.5.2.7	CIM_ComputerSystem.OperationalStatus.....	76
316	8.5.2.8	CIM_ComputerSystem.NameFormat.....	76
317	8.5.2.9	CIM_ComputerSystem.ElementName.....	76
318	8.5.2.10	CIM_ComputerSystem.IdentifyingDescriptions.....	77
319	8.5.2.11	CIM_ComputerSystem.OtherIdentifyingInfo.....	77
320	8.5.2.12	CIM_ComputerSystem.Caption.....	77
321	8.5.2.13	CIM_ComputerSystem.Description.....	77
322	9	IPMI Versioning Mapping.....	79
323	9.1	Introduction.....	79
324	9.2	Instance Diagrams.....	79
325	9.3	Mapping Requirements.....	80
326	9.4	CIM_RegisteredProfile Mapping.....	80
327	9.4.1	CIM_RegisteredProfile Methods.....	80
328	9.4.2	CIM_RegisteredProfile Properties.....	80
329	9.4.2.1	CIM_RegisteredProfile.InstanceID.....	81
330	9.4.2.2	CIM_RegisteredProfile.RegisteredOrganization.....	81
331	9.4.2.3	CIM_RegisteredProfile.OtherRegisteredOrganization.....	81
332	9.4.2.4	CIM_RegisteredProfile.RegisteredName.....	81
333	9.4.2.5	CIM_RegisteredProfile.RegisteredVersion.....	81
334	10	IPMI Entity Mapping.....	83
335	10.1	Introduction.....	83
336	10.2	Instance Diagrams.....	84

337	10.2.1 Example: System Board as Physical Parent.....	84
338	10.2.2 Example: Chassis as Physical Parent.....	85
339	10.3 Mapping Requirements	86
340	10.3.1 Mapping Entity ID's to CIM Classes.....	86
341	10.3.2 Entity Instantiation Algorithm	88
342	10.3.3 Entity Instance Correlation	95
343	10.4 CIM_PhysicalPackage Mapping.....	95
344	10.4.1 CIM_PhysicalPackage Methods	95
345	10.4.2 CIM_PhysicalPackage Properties	95
346	10.4.2.1 CIM_PhysicalPackage.Tag.....	96
347	10.4.2.2 CIM_PhysicalPackage.CreationClassName	96
348	10.4.2.3 CIM_PhysicalPackage.PackageType	97
349	10.4.2.4 CIM_PhysicalPackage.Manufacturer	99
350	10.4.2.5 CIM_PhysicalPackage.Model.....	99
351	10.4.2.6 CIM_PhysicalPackage.PartNumber	99
352	10.4.2.7 CIM_PhysicalPackage.SerialNumber.....	99
353	10.4.2.8 CIM_PhysicalPackage.Name	99
354	10.4.2.9 CIM_PhysicalPackage.HealthState	100
355	10.4.2.10 CIM_PhysicalPackage.OperationalStatus	100
356	10.4.2.11 CIM_PhysicalPackage.Description	100
357	10.4.2.12 CIM_PhysicalPackage.ElementName	100
358	10.5 CIM_Card Mapping.....	100
359	10.5.1 CIM_Card Methods	100
360	10.5.2 CIM_Card Properties.....	100
361	10.5.2.1 CIM_Card.Tag.....	102
362	10.5.2.2 CIM_Card.CreationClassName	102
363	10.5.2.3 CIM_Card.HostingBoard.....	102
364	10.5.2.4 CIM_Card.PackageType.....	102
365	10.5.2.5 CIM_Card.Manufacturer	104
366	10.5.2.6 CIM_Card.ManufacturerDate.....	104
367	10.5.2.7 CIM_Card.PartNumber.....	104
368	10.5.2.8 CIM_Card.SerialNumber.....	104
369	10.5.2.9 CIM_Card.Name	104
370	10.5.2.10 CIM_Card.HealthState	105
371	10.5.2.11 CIM_Card.OperationalStatus	105
372	10.5.2.12 CIM_Card.Description	105
373	10.5.2.13 CIM_Card.ElementName	105
374	10.6 CIM_Chip Mapping.....	105
375	10.6.1 CIM_Chip Methods	105
376	10.6.2 CIM_Chip Properties.....	105
377	10.6.2.1 CIM_Chip.Tag.....	106
378	10.6.2.2 CIM_Chip.CreationClassName	107
379	10.6.2.3 CIM_Chip.HealthState	107
380	10.6.2.4 CIM_Chip.OperationalStatus	107
381	10.6.2.5 CIM_Chip.Description	107
382	10.6.2.6 CIM_Chip.ElementName	107
383	10.7 CIM_Chassis Mapping.....	107
384	10.7.1 CIM_Chassis Methods.....	107
385	10.7.2 CIM_Chassis Properties	107
386	10.7.2.1 CIM_Chassis.Tag	109
387	10.7.2.2 CIM_Chassis.CreationClassName.....	109
388	10.7.2.3 CIM_Chassis.ChassisPackageType.....	109

389	10.7.2.4	CIM_Chassis.PartNumber	110
390	10.7.2.5	CIM_Chassis.SerialNumber	110
391	10.7.2.6	CIM_Chassis.HealthState	110
392	10.7.2.7	CIM_Chassis.OperationalStatus	110
393	10.7.2.8	CIM_Chassis.Description	110
394	10.7.2.9	CIM_Chassis.ElementName	110
395	10.8	CIM_Processor Mapping	110
396	10.8.1	CIM_Processor Methods	110
397	10.8.2	CIM_Processor Properties	110
398	10.8.2.1	CIM_Processor.SystemCreationClassName	112
399	10.8.2.2	CIM_Processor.CreationClassName	112
400	10.8.2.3	CIM_Processor.SystemName	112
401	10.8.2.4	CIM_Processor.DeviceID	112
402	10.8.2.5	CIM_Processor.UniqueID	112
403	10.8.2.6	CIM_Processor.EnabledState	112
404	10.8.2.7	CIM_Processor.Description	113
405	10.8.2.8	CIM_Processor.ElementName	113
406	10.8.2.9	CIM_Processor.HealthState	113
407	10.8.2.10	CIM_Processor.OperationalStatus	113
408	10.9	CIM_Fan Mapping	113
409	10.9.1	CIM_Fan Methods	113
410	10.9.2	CIM_Fan Properties	113
411	10.9.2.1	CIM_Fan.SystemCreationClassName	114
412	10.9.2.2	CIM_Fan.CreationClassName	114
413	10.9.2.3	CIM_Fan.SystemName	115
414	10.9.2.4	CIM_Fan.DeviceID	115
415	10.9.2.5	CIM_Fan.Description	115
416	10.9.2.6	CIM_Fan.ElementName	115
417	10.9.2.7	CIM_Fan.HealthState	115
418	10.9.2.8	CIM_Fan.OperationalStatus	115
419	10.10	CIM_PowerSupply Mapping	115
420	10.10.1	CIM_PowerSupply Methods	115
421	10.10.2	CIM_PowerSupply Properties	116
422	10.10.2.1	CIM_PowerSupply.SystemCreationClassName	116
423	10.10.2.2	<"CIM_ComputerSystem" or the name of the subclass created>	
424		CIM_PowerSupply.CreationClassName	116
425	10.10.2.3	CIM_PowerSupply.SystemName	116
426	10.10.2.4	CIM_PowerSupply.DeviceID	117
427	10.10.2.5	CIM_PowerSupply.Description	117
428	10.10.2.6	CIM_PowerSupply.ElementName	117
429	10.10.2.7	CIM_PowerSupply.HealthState	117
430	10.10.2.8	CIM_PowerSupply.OperationalStatus	117
431	10.11	CIM_Memory Mapping	117
432	10.11.1	CIM_Memory Methods	117
433	10.11.2	CIM_Memory Properties	117
434	10.11.2.1	CIM_Memory.SystemCreationClassName	119
435	10.11.2.2	CIM_Memory.CreationClassName	119
436	10.11.2.3	CIM_Memory.SystemName	119
437	10.11.2.4	CIM_Memory.DeviceID	119
438	10.11.2.5	CIM_Memory.Description	120
439	10.11.2.6	CIM_Memory.ElementName	120
440	10.11.2.7	CIM_Memory.HealthState	120

441	10.11.2.8	CIM_Memory.OperationalStatus	120
442	10.12	CIM_Battery Mapping	120
443	10.12.1	CIM_Battery Methods	120
444	10.12.2	CIM_Battery Properties	120
445	10.12.2.1	CIM_Battery.SystemCreationClassName	122
446	10.12.2.2	CIM_Battery.CreationClassName	122
447	10.12.2.3	CIM_Battery.SystemName.....	122
448	10.12.2.4	CIM_Battery.DeviceID.....	122
449	10.12.2.5	CIM_Battery.BatteryStatus.....	122
450	10.12.2.6	CIM_Battery.Description	122
451	10.12.2.7	CIM_Battery.ElementName	123
452	10.12.2.8	CIM_Battery.HealthState	123
453	10.12.2.9	CIM_Battery.OperationalStatus	123
454	10.13	CIM_DiskDrive Mapping	123
455	10.13.1	CIM_DiskDrive Methods.....	123
456	10.13.2	CIM_DiskDrive Properties	123
457	10.13.2.1	CIM_DiskDrive.SystemCreationClassName.....	124
458	10.13.2.2	CIM_DiskDrive.CreationClassName	125
459	10.13.2.3	CIM_DiskDrive.SystemName	125
460	10.13.2.4	CIM_DiskDrive.DeviceID.....	125
461	10.13.2.5	CIM_DiskDrive.Description.....	125
462	10.13.2.6	CIM_DiskDrive.ElementName	125
463	10.13.2.7	CIM_DiskDrive.HealthState.....	125
464	10.13.2.8	CIM_DiskDrive.OperationalStatus.....	125
465	11	IPMI Watchdog Mapping	126
466	11.1	Introduction	126
467	11.2	Instance Diagrams	126
468	11.3	Mapping Requirements	127
469	11.4	CIM_Watchdog Mapping	127
470	11.4.1	CIM_Watchdog Methods	127
471	11.4.1.1	Method: Reset()	127
472	11.4.1.2	Method: RequestStateChange.....	128
473	11.4.1.3	Method: KeepAlive	128
474	11.4.2	CIM_Watchdog Properties	129
475	11.4.2.1	CIM_Watchdog.SystemCreationClassName.....	130
476	11.4.2.2	CIM_Watchdog.SystemName	131
477	11.4.2.3	CIM_Watchdog.CreationClassName	131
478	11.4.2.4	CIM_Watchdog.DeviceID.....	131
479	11.4.2.5	CIM_Watchdog.MonitoredEntity.....	131
480	11.4.2.6	CIM_Watchdog.MonitoredEntityDecription.....	132
481	11.4.2.7	CIM_Watchdog.TimeoutInterval	132
482	11.4.2.8	CIM_Watchdog.TimerResolution	132
483	11.4.2.9	CIM_Watchdog.TimeOfLastExpiration	132
484	11.4.2.10	CIM_Watchdog.MonitoredEntityOnLastExpiration	132
485	11.4.2.11	CIM_Watchdog.ActionOnExpiration.....	133
486	11.4.2.12	CIM_Watchdog.EnabledState	133
487	11.4.2.13	CIM_Watchdog.DefaultState	133
488	11.4.2.14	CIM_Watchdog.RequestedState.....	134
489	11.4.2.15	CIM_Watchdog.Name.....	134
490	11.4.2.16	CIM_Watchdog.OperationalStatus.....	134
491	11.4.2.17	CIM_Watchdog.HealthState.....	134
492	11.4.2.18	CIM_Watchdog.LogState	135

493	11.4.2.19	CIM_Watchdog.ActionOnPretimeoutInterval	135
494	11.4.2.20	CIM_Watchdog.PretimeoutInterval	135
495	12	IPMI Software Identity Mapping	136
496	12.1	Introduction	136
497	12.2	Instance Diagrams	136
498	12.3	Mapping Requirements	137
499	12.4	CIM_SoftwareIdentity Mapping	137
500	12.4.1	CIM_SoftwareIdentity Methods	137
501	12.4.2	CIM_SoftwareIdentity Properties	137
502	12.4.2.1	CIM_SoftwareIdentity.InstanceID	138
503	12.4.2.2	CIM_SoftwareIdentity.RevisionNumber	139
504	12.4.2.3	CIM_SoftwareIdentity.VersionString	139
505	12.4.2.4	CIM_SoftwareIdentity.Manufacturer	139
506	12.4.2.5	CIM_SoftwareIdentity.Name	139
507	12.4.2.6	CIM_SoftwareIdentity.OperationalStatus	139
508	12.4.2.7	CIM_SoftwareIdentity.HealthState	139
509	13	Acknowledgments	140
510		End of Document	140
511			

512

List of Figures

513	Figure 1	Class Diagram – IPMI Sensor Profile Mapping	20
514	Figure 2	Instance Diagram – IPMI Sensor Profile Mapping	21
515	Figure 3	Class Diagram - IPMI Log	40
516	Figure 4	Instance Diagram – IPMI Log	41
517	Figure 5	Class Diagram – IPMI Computer System Power Management	55
518	Figure 6	Instance Diagram – IPMI Computer System Power Management	56
519	Figure 7	Class Diagram – IPMI User ID Management	61
520	Figure 8	Instance Diagram – IPMI User ID Management	62
521	Figure 9	Class Diagram – IPMI BMC MAP Management	69
522	Figure 10	Instance Diagram – IPMI BMC MAP Management	70
523	Figure 11	Class Diagram – IPMI Versioning	79
524	Figure 12	Instance Diagram – IPMI Versioning	80
525	Figure 13	Class Diagram – IPMI Physical Entities	83
526	Figure 14	Class Diagram – IPMI Physical Associations	84
527	Figure 15	Instance Diagram – IPMI Entities (System Board Parent)	85
528	Figure 16	Instance Diagram – IPMI Entities (Chassis Parent)	86
529	Figure 17	“Present” and “Absent” Entity Association Hierarchies	89
530	Figure 18	Class Diagram – IPMI Watchdog	126
531	Figure 19	Instance Diagram – IPMI Watchdog	127
532	Figure 20	Class Diagram – Software Identity	136
533	Figure 21	Instance Diagram – Software Identity	137

534

535

List of Tables

536	Table 1 CIM_Sensor Class Properties.....	23
537	Table 2 CIM_NumericSensor Class Properties.....	35
538	Table 3 CIM_RecordLog Class Properties.....	43
539	Table 4 CIM_LogRecord Class Properties.....	46
540	Table 5 CIM_CommandService Class Properties.....	53
541	Table 6 CIM_PowerManagementService Class Properties.....	58
542	Table 7 CIM_PowerManagementCapabilities Class Properties.....	59
543	Table 8 CIM_AssociatedPowerManagementService Class Properties.....	59
544	Table 9 CIM_Account Class Properties.....	64
545	Table 10 CIM_Group Class Properties.....	66
546	Table 11 CIM_AuthorizedPrivilege Class Properties.....	67
547	Table 12 CIM_AdminDomain Class Properties.....	71
548	Table 13 CIM_ComputerSystem Class Properties.....	73
549	Table 14 CIM_RegisteredProfile Class Properties.....	81
550	Table 15 IPMI Entity ID to CIM Class Mapping.....	88
551	Table 16 “Physical” Sensor Types to Entity Mapping.....	92
552	Table 17 FRU - Entity Instantiation.....	92
553	Table 18 CIM_PhysicalPackage Class Properties.....	95
554	Table 19 CIM_Card Class Properties.....	101
555	Table 20 CIM_Chip Class Properties.....	106
556	Table 21 CIM_Chassis Class Properties.....	108
557	Table 22 CIM_Processor Class Properties.....	111
558	Table 23 CIM_Fan Class Properties.....	114
559	Table 24 CIM_PowerSupply Class Properties.....	116
560	Table 25 CIM_Memory Class Properties.....	118
561	Table 26 CIM_Battery Class Properties.....	121
562	Table 27 CIM_DiskDrive Class Properties.....	123
563	Table 28 CIM_Watchdog Class Properties.....	130
564	Table 29 CIM_SoftwareIdentity Class Properties.....	138
565			

566 1 Introduction

567 Note on providing feedback: Feedback or comments on this IPMI CIM Mapping Guideline
568 should be sent to: jon_hass@Dell.com, tom.slaight@intel.com, steve.lyle@hp.com, and
569 steffen.hulegaard@avocent.com. Any feedback provided must be done on a royalty free
570 licensing basis. For feedback to be considered and incorporated into this IPMI CIM Mapping
571 Guideline, the submitter must agree in writing to the IPMI CIM Mapping Feedback Agreement.
572 See the directions in the Copyright/IP Disclaimer above.

573 The purpose of the IPMI-CIM Mapping Guideline is to ensure interoperability in the usage of
574 CIM classes for (primarily) server management. Additionally, the IPMI-CIM Mapping
575 Guideline defines a consistent way to connect service processors (SPs) that implement IPMI
576 compliant control interfaces to CIM based control interfaces. Such service processors (SPs) are
577 often called baseboard management controllers (BMCs).

578 1.1 Target Audience

579 The IPMI CIM Mapping Guideline is intended for use by developers of IPMI-to-CIM
580 translation/mapping logic.

581 1.2 Conventions

582 The key phrases and words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT,
583 SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to
584 be interpreted as in RFC 2119.

585 1.3 Reference Documents

586 [1] Common Information Model (CIM) Specification, V2.2, June 14, 1999 - Downloadable from
587 <http://www.dmtf.org/spec/cims.html>

588 [2] Unified Modeling Language (UML) from the Open Management Group (OMG) -
589 Downloadable from <http://www.omg.org/uml/>>

590 [3] Intelligent Platform Management Interface (IPMI) Specification, February 20, 2002 -
591 Downloadable from <http://www.intel.com/design/servers/ipmi/>

592 [IPMI 1.5] Refers to the *Intelligent Platform Management Interface Specification v1.5, revision 1.1*,
593 February 20, 2002. Copyright © 2002 Intel Corporation, NEC Corporation, Dell
594 Computer Corporation, and Hewlett-Packard Company, plus the corresponding errata and
595 addenda document [IPMI Errata], referenced below.

596
597 [IPMI 2.0] Refers to the combination of two documents: *Intelligent Platform Management*
598 *Interface Specification - Second Generation, v2.0*, revision 1.0, February 2, 2004.
599 Copyright © 2004 Intel Corporation, Hewlett-Packard Company, NEC
600 Corporation, Dell Computer Corporation. Plus: [IPMI Errata] Available from
601 <http://www.intel.com/design/servers/ipmi/>

602 [IPMI Errata] Intelligent Platform Management Interface Second Generation Specification,
 603 v2.0, revision 1.0, Intelligent Platform Management Interface Specification v1.5,
 604 revision 1.1, Addendum Document, Revision 1, June 1, 2004. Copyright © 2004
 605 Intel Corporation, Hewlett-Packard Company, NEC Corporation, Dell Computer
 606 Corporation. Available from <http://www.intel.com/design/servers/ipmi>

607 [CIM 2.9] DMTF CIM 2.9 Schema, Copyright © 2005 Distributed Management Task Force,
 608 Inc. (DMTF). All rights reserved.

609 [CIM 2.10] DMTF CIM 2.10 Schema, Copyright © 2005 Distributed Management Task
 610 Force, Inc. (DMTF). All rights reserved.

611

612 **1.4 Terminology**

613

Term	Definition
in-band	Dependent upon either hardware and/or software resources used for supporting the principal workload.
in-service	Dependent on software, services and/or interfaces that require a concurrently running operating system.
out-of-band	Independent of both hardware and/or software resources used for supporting the principal workload.
out-of-service	Independent of software, services and/or interfaces that require a concurrently running operating system.
SM-Bus	System Management Bus

614

615 **1.5 Acronyms and Abbreviations**

616

Acronym	Definition
BMC	Baseboard Management Controller
BT	Block Transfer (in-band IPMI interface)
CLP	Command Line Protocol (from the DMTF WBEM/CIM SMASH initiative)
DMTF	Distributed Management Task Force (www.dmtf.org)
I2C	Inter-Integrated-Circuit bus (a 2/3 wire bus for certain in-band IPMI interfaces)
IB	In-band (see Terminology section)
ICMB	Intelligent Chassis Management Bus (part of IPMI)
IPMB	Intelligent Platform Management Bus (part of IPMI)
IPMI	Intelligent Platform Management Interface (www.intel.org/design/servers/ipmi)
KCS	Keyboard Controller Style (in-band IPMI interface)
LAN	Local Area Network
MAP	Manageability Access Point
OOB	Out-of-Band (see Terminology section)
PCI	Peripheral Component Interconnect (Bus - http://www.webopedia.com/TERM/P/PCI.html)
SDR	Sensor Data Repository (defined by IPMI)
SDRs	Sensor Data Records (which are resident in the IPMI SDR)
SMASH	Systems Management Architecture for Server Hardware
SMIC	System Management Interface Chip (Interface – an in-band IPMI interface)
SP	Service Processor
SSIF	SM-BUS System Interface (an in-band IPMI interface)
WBEM	Web Based Enterprise Management (Initiative within the DMTF)

617 1.6 Editorial Conventions

618 By default, we'll use IPMI terms in each property/method mapping section. If you see a term
619 like 'SensorType', this means the IPMI SensorType (not the CIM SensorType). If we need to
620 refer to a CIM property/method, we will prefix with the corresponding CIM class name and a
621 'dot' separator. Here is an example:

622 `CIM_Sensor.SensorType`

623 Sometimes, a CIM property/method reference will be scoped to a specific specialization rather
624 than CIM base class that introduces a particular property and/or method. What follows is an
625 example of such a specialization scoped reference

626 `CIM_NumericSensor.SensorType`

627 • Editorial Shorthand

628

Notation	Meaning
<code>:=</code>	"Defined as" or "Defined to be"
<code>=></code>	Maps to

629 **1.7 Notational Conventions**630 • **Operators**

631

Operator	Example	Description
<>	<SDR Type 1 bytes 49+> <CIM_Sensor.Name>	<p>These brackets are a substitution macro. The <...> bracket enclosed text references something whose corresponding value should be substituted. For example, Table 37-1 in the IPMI v1.5 specification describes bytes 49+ in the Type 1 IPMI Sensor Data Record (SDR) as being the sensor's ID string bytes.</p> <p>Reuse and the <i>Define Once Discipline</i>: The <CIM_Sensor.Name> example references the elsewhere defined mapping for the CIM_Sensor class's Name property. In the <CIM_Sensor.Name> example, the previously defined mapping is being referenced and re-used (rather than defined in more than one place). We especially encourage such reuse when references cross documents and/or chapter divisions.</p>
(<type>)	(Boolean) (String)	<p>Cast/convert to the given fundamental type.</p> <p>For example, a binary value [0,1] might be casted/converted to a CIM boolean value ["true","false"]. We often use the cast operator to emphasize the types of the values involved. These redundant/unnecessary casts (of say a string value to a string value) are simply of way of making the mapping specification more self-documenting.</p>
EStr()	EStr(SensorType)	<p>Maps a value to an associated description string. This is the "explain" string macro/function. It amounts to mapping an enumeration's value to an associated enumeration's symbolic value. This also covers any other similar sort of associative array based mapping-to-string. For example, IPMI v1.5 Sensor Type (hex defined) numeric value is defined in Table 36-3. Each value is associated in Table 36-3 with a corresponding description string.</p>
TruncAt(<integer>, string)	TruncAt(5, "Hereisabigstring") == "Herei"	<p>Truncates a sequence at a specified number of entries. Often used to truncate a string (a sequence of characters) to a given length. If the string is shorter than the given length, TruncAt indicates no change to the input string (i.e. a no-op).</p>
Relops (=,>,<,etc.) LogicalOps (&&,)	SensorType=02h	<p>Relational operators represent filtering/scoping predicates. For example, SensorType=[01h-04h] means SensorType values limited to the range 01h to 04h. Notice that this mixes the equivalence operator and a regular expression. The same example could be denoted as follows:</p> <p>SensorType>=01h && SensorType<=04h</p> <p>Or</p> <p>SensorType>00h && SensorType<05h</p>

	"foo" "bar" => "foobar"	String concatenation. The ' ' operator is a logical 'OR' if it's operands are relational expressions. It is string concatenation otherwise (and it requires string type operand context to function as string concatenation).
--	-------------------------------	--

632
633

634 • **Regular expressions**

635 Some common regular expression syntax may be intermingled in the mapping
636 definitions.

637 If we eventually need both a rigorously defined and an advanced regular expression
638 syntax, our direction is to conform the Python "re" module regular expression syntax
639 (see Programming Python, 2nd Edition, Lutz, Mark, O'Reilly & Associates, Inc,
640 Sebastopol, California, 2001, ISBN 0-596-00085, p1041+). The Python regular
641 expression syntax deliberately subsumes the Perl regular expression syntax.

642 • **Python style pseudo-code (when and if necessary)**

643 Since Python was designed to be a C/C++ style pseudo-code, we'll adopt the Python
644 language conventions for any procedural mapping specifications. We do not intend to
645 write executable Python code. We do not imply anything about Python nor Python's
646 suitability for any given purpose. Our interest in Python is only as a means to express
647 any necessary IPMI-to-CIM procedural mapping logic (fragments). We adopt Python's
648 widely familiar, C/C++ style, syntax . This is just an expedient choice of procedural
649 specification languages.

650

651 When, and if, we need to resort to procedural mapping specifications, we'll introduce
652 the corresponding Python fragments with a "*Procedure*" heading. The Python-like
653 pseudo-code will indulge in some non-Python notation where the possibility for
654 confusion is low and where the notation reinforces connections to either IPMI
655 terminology/notation and/or CIM terminology/notation. For example, IPMI constant
656 notation like "03h" might be used rather than Python's "0x3". We also use compact
657 self-assignment forms from C/C++ (like ||= for concentration assignment). This is just
658 more concise and often more convenient.

659 • **In-line comments**

660 In-line (pseudo-code) comment syntax follows the Python conventions.

661 # This is a comment until end of current line. Multi-line comments

662 # much use a '#' character to introduce each comment line.

663 **1.8 Referenced Standards**

Standard/Specification	Version	Source	Citation
CIM Infrastructure Specification	2.3	DMTF	DSP 0004, www.dmtf.org/standards/cim
CIM Schema	2.10	DMTF	www.dmtf.org/standards/cim

Standard/Specification	Version	Source	Citation
CIM Specification	2.2.2	DMTF	DSP 0007, www.dmtf.org/standards/cim
IPMI Specification	1.5	Intel	www.intel.org/design/servers/ipmi
IPMI Specification	2.0	Intel	www.intel.org/design/servers/ipmi

664

665 **2 Common Steps for Mapping Process**

666 This section presents the general process that instrumentation provider software would use for
667 accessing an IPMI-based platform management subsystem, discovering its available
668 management capabilities, and mapping those capabilities to CIM objects described in this
669 document. These steps are presented as a guide to software design and to applying the IPMI
670 specifications. These steps are summarized as follows:

- 671 1. Discover and connect to the IPMI interface for the particular managed system.
- 672 2. Discover and enumerate the IPMI Entities, Sensors, and FRU information creating a
673 logical ‘list’ of IPMI capabilities.
- 674 3. Take this logical listing and use it to instantiate and populate CIM objects and
675 associations according to the discovered capabilities on the given system.

676 **2.1 Discovering IPMI Interfaces**

677 Instrumentation that maps from IPMI to CIM must first discover and connect to the IPMI
678 platform management subsystem via one of the IPMI interfaces. The IPMI interfaces may be
679 classified into three types: System, Remote, and Internal.

680 **2.1.1 System Interfaces**

681 The System Interface is a set of registers that software running on the managed system uses to
682 directly access the IPMI subsystem. There are three types of system interface defined in [IPMI
683 1.5]: the KCS (keyboard controller style) interface, the BT (block transfer) interface, and the
684 SMIC (server management interface chip) interface. Of these, the KCS interface is by far the
685 most widely deployed, followed by the BT interface. The SMIC interface is an older interface
686 that is not recommended for new designs. [IPMI 2.0] introduced a fourth system interface, called
687 SSIF (SMBus System Interface) that is used with low cost IPMI BMCs (baseboard management
688 controllers).

689 Local instrumentation software must decide which types of System Interface it will support. It is
690 highly recommended that the KCS interface be supported. Other interfaces should be considered
691 based on the target systems for the software. The IPMI Web Site,
692 <http://www.intel.com/design/servers/ipmi>, has reference drivers and sample code that can assist
693 with the development of software for a number of the system interfaces.

694 Section 6 of the IPMI specifications describes the System Interfaces and their operation. In this
695 section, there is a subsection titled “System Interface Discovery and Multiple Interfaces”. This
696 subsection describes the steps that software should use to discover the type of system interface
697 used on a given platform, its location, and how to handle the situation if multiple system
698 interfaces are present.

699 **2.1.2 Remote Interfaces**

700 The IPMI platform management subsystem on a given system may also be accessible remotely
701 via IPMI remote interfaces. The remote interfaces include serial/modem and LAN (Ethernet).

702 The IPMI remote interfaces incorporate their own media-specific protocols for delivering IPMI
703 messages to the platform management subsystem. However, the IPMI messages themselves are
704 the same regardless of the interface over which they're delivered. Thus, in general, the same
705 IPMI management capabilities that can be accessed locally can also be accessed via the remote
706 interfaces³. This means that the same general process of mapping IPMI capabilities to CIM
707 locally (via the system interface) can also be used for mapping IPMI to CIM via the remote
708 interfaces. This enables the possibility of creating 'proxy' software to provide a CIM-based
709 interface to remote IPMI-based systems.

710 Because the interfaces provide remote access, the serial/modem and LAN interfaces incorporate
711 options for securing the interfaces and restricting access to specific users that have been
712 configured and enabled for IPMI access to a particular BMC. Thus, a remote application must
713 not only support the protocol requirements for deliver IPMI messages over the interface, but also
714 include support for 'logging in' to the remote system as a user that has the necessary privilege
715 level to support IPMI-to-CIM mapping.

716 Unless otherwise indicated, the full set of capabilities enabled by IPMI-to-CIM mapping can be
717 accomplished by a user that has "Operator" privilege. Refer to Appendix G in the IPMI
718 specifications for a listing of the privilege levels for the IPMI commands.

719 IPMI over Serial/Modem has several different access protocols that can be used. These are
720 referred to as "Basic Mode", "Terminal Mode", and "PPP Mode". Basic Mode support is
721 mandatory for IPMI over Serial/Modem, the other modes are optional. An application can
722 therefore rely on Basic Mode as being common across platforms that support IPMI over
723 Serial/Modem.

724 The approach to handling user logins varies between the serial access protocols. However, all
725 authenticated interfaces share the same association of user names and privilege levels. The PPP
726 interface also incorporates the option for an additional layer of "link" security where the serial
727 communication connection itself must first be authenticated, and then IPMI users are
728 authenticated on top of that. An application that supports IPMI over PPP should also support the
729 options for link-level security.

730 IPMI over LAN includes support for enabling applications to discover IPMI-based systems on
731 the network. The serial/modem interfaces presume that the application has a-priori knowledge of
732 the phone number necessary to connect to the managed system, or that there is a direct
733 connection available between the managing (console) system and the IPMI serial interface.
734 However, once the serial connection is established, the interfaces include mechanisms that allow
735 the availability of IPMI to be confirmed.

736 It should be noted that the serial interfaces may also be used with a capability called "Serial Port
737 Sharing" where access to the system's serial controller can be shared with IPMI access. This
738 guideline does not cover CIM-based instrumentation access to the system serial connection via
739 IPMI Serial Port Sharing. A remote application should be aware, however, that it will need to
740 take additional steps to switch the serial connection to the BMC if Serial Port Sharing is in effect.

³ Typically, only certain interface-specific IPMI commands will not be available on other interfaces. For example, commands related to LAN discovery and session setup are not available over other interfaces.

741 **2.2 Internal Interfaces**

742 IPMI also defines interfaces that can be used to enable other management controllers, add-in
743 cards, and other devices to access IPMI ‘inside the box’. These internal interfaces include the
744 IPMB (Intelligent Platform Management Bus), the PCI SMBus, and an inter-chassis bus referred
745 to as the ICMB⁴ (Intelligent Chassis Management Bus). (While the ICMB bus extends between
746 chassis, it is more characteristic of an internal interface because it is considered to be a
747 ‘physically secure’ and therefore does not include ‘user logins’ for controlling user access.

748 As with the remote interfaces, the same IPMI management capabilities that can be accessed via
749 the System Interface can also be accessed via the internal interfaces. This means that the same
750 general process of mapping IPMI capabilities to CIM can be used, creating the possibility of an
751 add-in card that incorporates the IPMI-to-CIM mapping to provide a CIM-based remote interface
752 for the managed system.

753 **2.3 Discovery and Enumeration of IPMI Entity, Sensor, and FRU** 754 **Information and Associations**

755 Once software is able to access IPMI, the next major step of the mapping process is for software
756 to discover and enumerate the particular platform’s IPMI capabilities so they can be mapped into
757 CIM. This includes discovering the control capabilities, sensors, FRU (field replaceable unit)
758 information, and the associations between sensors and FRU information for the entities that are
759 monitored and managed through IPMI.

760 This discovery and enumeration process can be thought of as creating a logical ‘list’ of the IPMI
761 capabilities for a given platform. Once this logical listing is created, it can be used to drive the
762 instantiation of CIM objects for accessing those capabilities.

763 **2.3.1 Recommendations on the access and use of SDR Info**

764 The IPMI Sensor Data Records (SDRs) are a key element to the enumeration and discovery of
765 IPMI sensors, FRU information, and the entities they are associated with. In most cases, it will
766 be faster and more efficient for software to read out the entire set of sensor data records in bulk
767 into a local file or system RAM and work with the copy of the SDRs rather than to access SDRs
768 and process them one at a time.

769 In addition, if an application is responsible for monitoring multiple systems, it is useful to cache
770 information about the managed systems rather than accessing it anew each time the managed
771 system is accessed. IPMI provides timestamps on the SDR Repository (accessed using the *Get*
772 *SEL Info* command) that software can use to tell if the SDR contents have changed since the last
773 time the info was accessed.

⁴ There is a separate document that describes the ICMB and its operation. Refer to the reference to [ICMB] in the IPMI specifications.

774 **2.4 Entity Discovery**

775 IPMI supports a type of SDR called “Entity Association Records”. These records allow physical
776 or logical Entities to grouped together under a common ‘container’ Entity. The container Entity
777 may itself be a ‘contained’ Entity in another Entity Association. Thus, Entity Associations can
778 form a ‘tree’ hierarchy expressing relationships between Entities and/or groups of Entities.

779 There can be an interdependency between IPMI Sensors and Entities. If the container Entity at
780 the top of a given Entity Association or Entity Association Hierarchy is absent (as indicated by a
781 Presence sensor or by presence bit associated with a sensor for the Entity) then all ‘contained’
782 Entities under that hierarchy are considered to be absent. Consequently, any Sensors associated
783 with those contained Entities are also considered absent.

784 Because multiple sensors can be associated with a single Entity, and because the presence of
785 multiple Entities can be determined by a the presence of an Entity Association or Entity
786 Association Hierarchy, it is generally more efficient to determine the presence of Entities first,
787 then Sensors.

788 Whether an Entity Association is ‘present’, or ‘absent’ is based on whether the container Entity
789 for the Association has sensors associated with it, and if so, what the state of those sensors is, as
790 follows:

- 791 • If the container Entity does not have any sensors associated with it, then the association
792 is assumed to be ‘present’.
- 793 • If the container Entity has one or more sensors associated with it, but all sensors are
794 ‘scanning disabled’ or are inaccessible, then the association should be considered
795 ‘absent’ and should be ignored. I.e. software should act as if that association was not
796 even listed in the SDRs.
- 797 • If the container Entity has one or more ‘scanning enabled’ associated with it, and none
798 of those sensors are a presence sensor or presence bit that indicates that the container
799 Entity is absent, then the association is assumed to be ‘present’ (valid).
- 800 • If the container Entity has a presence sensor or bit associated with it that indicates the
801 container Entity is absent, then the overall association is said to be *‘explicitly absent’*
802 and the container Entity and any contained Entities and lower level associations should
803 not be instantiated because of being listed under that association. (Note that the same
804 Entities may still become instantiated because they’re listed under a different Entity
805 association or association hierarchy that *is* present.)

806 **2.4.1 Entity presence rules**

807 The following is an brief overview of the different ways Entity presence can be determined in
808 IPMI. This is described in more detail in 10.3.2 Entity Instantiation Algorithm. In this section,
809 the words “the Entity” shall be used as a shorthand for “The particular instance and type of
810 Entity”.

811 **2.4.1.1 Explicit by Entity Association**

812 The Entity listed under an “Entity Association” that is defined by one or more Entity Association
813 Records and is present

814 AND

815 None of the Entity Associations that are higher up in the Entity hierarchy are explicitly absent.

816 AND

817 The Entity is not listed in the Entity ID/Entity Instance fields of the SDR for a Presence sensor,
818 nor a sensor that includes a presence bit that indicates the Entity is ‘not present’ or ‘not installed’
819 (see note below).

820 **2.4.1.2 Explicit by Sensor SDR or Event SDR**

821 The Entity is not listed under *any* present or explicitly absent Entity Association Hierarchy, but is
822 listed in the Entity ID/Entity Instance fields of one or more “sensor” or “event only” SDRs
823 (Type 01h, 02h, or 03h).

824 AND

825 The Entity is not listed in the Entity ID/Entity Instance fields of the SDR for a Presence sensor,
826 nor a sensor that includes a presence bit that indicates the Entity is ‘not present’ or ‘not
827 installed’(see note below).

828 AND

829 At least one sensor associated with then Entity is enabled (See 2.5, Sensor Discovery, for the
830 definition of an ‘enabled’ sensor).

831 **2.4.1.3 Explicit by FRU Device Locator Record**

832 The Entity is not listed under *any* Entity Association Hierarchy (regardless of whether the
833 hierarchy is present or absent), but is listed in the Entity ID/Entity Instance fields of one or more
834 FRU Device Locator SDRs (Type 11h).

835 AND

836 There are no sensors associated with the Entity, or at least one sensor associated with then Entity
837 is enabled (See 2.5, Sensor Discovery, for the definition of an ‘enabled’ sensor).

838 AND

839 The Entity is not listed in the Entity ID/Entity Instance fields of the SDR for a Presence sensor,
840 nor a sensor that includes a presence bit that indicates the Entity is ‘not present’ or ‘not installed’
841 (see note below).

842 AND

843 The FRU Device for associated with the Entity is accessible by software.

844 **2.4.1.4 Implicit by Sensor Type**

845 The existence of an Entity can also be inferred by the Sensor Type. For example, if a Fan sensor
846 is present, then it implies the existence of a Fan. This is done only when the Sensor Type and the
847 Entity ID do not match, and only for certain Sensor Types, listed in a table in Section 10.3.1,
848 Mapping Entity ID's to CIM Classes. An Entity that has its presence inferred this way is not
849 explicitly identified by IPMI Entity ID/Entity Instance numbers. Therefore, for instance
850 identification purposes, some unique ID must be synthesized. In this specification, a combination
851 of the Sensor Number, Owner ID, and other information, is used to synthesize a CIM
852 Sensor.DeviceID value that is used as the instance identifier.(see note below). Thus, Entity can
853 be inferred to exist if:

854 The Entity the is explicitly associated with the sensor (the Entity identified by the Entity ID and
855 Entity Instance fields in the SDR) is present by one of the two preceding explicit presence
856 mechanisms.

857 AND

858 The Sensor Type in the Sensor Data Record is of a type that implies the existence of a physical
859 Entity, per Table 16 "Physical" Sensor Types to Entity Mapping .

860 AND

861 The Entity ID implied by the Sensor Type in the Sensor Data Record for the sensor does not
862 match the explicit type given by the Entity ID in the Sensor Data Record.

863 AND

864 At least one sensor associated with then Entity is enabled (See 2.5, Sensor Discovery, for the
865 definition of an 'enabled' sensor).

866 **2.4.1.5 Implicit by FRU Data**

867 The existence of an Entity can also be inferred by IPMI FRU data. For example, if there is
868 Chassis information in a FRU Device, it implies the existence of a physical chassis. This
869 mechanism is used when there is not a FRU Device Locator record for the FRU Device that calls
870 out a particular Entity ID and Entity Instance for an Entity to be associated with the FRU data.
871 As with Entities whose existence is "Implicit by Sensor Type" it is necessary to synthesize an
872 instance identifier in lieu of having explicit Entity ID and Entity Instance information.

873 FRU Information Type:

- 874 • Chassis Info Area, implies existence of a Chassis Entity.
- 875 • Board Info Area, implies existence of a board or module.

876

877 Refer to the steps outlined in Section 10, IPMI Entity Mapping, and Table 16 “Physical”
878 Sensor Types to Entity Mapping.

879 **Note:**

880 The two presence mechanisms (Presence sensor or presence bit) should not be used
881 simultaneously in an implementation. However, if an implementation does utilize both
882 mechanisms with a given Entity, the following interpretation rule applies

- 883 • the Entity should be considered to be 'not present' (absent) if either mechanism reports
884 'not present'.
- 885 • the Entity should be considered to be 'present' if both mechanisms report present
- 886 • if scanning is disabled for the sensor for either explicit mechanism, the remaining
887 mechanism is utilized as if the other didn't exist.
- 888 • if scanning is disabled for both mechanisms, the rules for the Entity Presence Sensor
889 apply - meaning that ALL sensors associated with that Entity should be ignored.
890

891 Refer to the IPMI Specifications to find out which sensor types include sensor-specific offset bits
892 that serve as presence bits for associated Entities.

893 **2.5 Sensor Discovery**

894 This step consists of the discovery of ‘enabled’ sensors or ‘event only’ sensors where the Entity
895 associated with the sensor is part of an Entity Association or Entity Association Hierarchy that is
896 present, or the Entity is not covered by an Entity Association or Entity Association Hierarchy.

897 An ‘enabled’ sensor is a sensor that responds to a *Get Sensor Reading*, *Get Sensor Event Enable*,
898 or *Get Sensor Event Status* that returns bit [6] as 1b, indicating ‘sensor scanning enabled’. An
899 ‘event only’ sensor is a sensor that is listed using the ‘type 03h’ Event Only SDR.

900 The Entity that is associated with a sensor is identified by the Entity ID and Entity Instance fields
901 in the SDR for the sensor.

902 If a sensor is not listed in the SDRs it should be ignored. I.e. should be considered ‘non-existent’
903 with respect to this Sensor Discovery step.

904 **2.5.1 Sensor Presence Rules**

905 Thus, as sensor should be considered to be present (and a candidate for CIM instantiation) if:

- 906
- 907 The sensor is listed in the SDRs
- 908 AND
- 909 The sensor is ‘scanning enabled’
- 910 AND

911 The Entity associated with the sensor is not part of an Entity Association or Entity
912 Association Hierachy

913
914 OR

915 The sensor is listed in the SDRs
916 AND

917 The sensor is ‘scanning enabled’
918 AND

919 The Entity associated with the sensor is part of an Entity Association or Entity
920 Association Hierachy that is present (refer to 2.4.1, Entity presence rules, above).
921

922 **2.6 FRU Device Discovery**

923 IPMI FRU Devices hold FRU (Field Replaceable Unit) information for different replaceable
924 Entities in the system. This can include information such as serial number, model, manufacturer,
925 part number, etc. FRU Devices are accessed by software either as ‘logical’ or ‘physical’ FRU
926 Devices. Logical FRU Devices are accessed by sending *Read-* and *Write FRU Data* IPMI
927 command to the management controller that provides access to the device.

928 Physical FRU Devices (also referred to as ‘non-intelligent’ FRU Devices) are accessed as
929 separate SEEPRAM (Serial Electrically Erasable Programmable Read-Only Memory) devices on
930 a physical I²C or SMBus segment. Thus, physical FRU devices are often used provide FRU
931 information on add-in cards, since only a simple SEEPRAM is required to provide the
932 information, rather than a management controller that can handle IPMI commands. To access
933 physical FRU devices, software uses the IPMI *Master Write-Read* command to generate
934 SEEPRAM access transactions on the target I²C or SMBus behind a management controller.

935 Logical and Physical FRU Device access is described in more detail in the IPMI Specification. In
936 particular, refer to the chapter 38, Accessing FRU Devices, and the documentation on the *Read-*
937 and the *Master Write-Read* command.

938 FRU Devices are discovered by the following mechanisms:

939 **2.6.1 By BMC Primary FRU Device**

940 Since IPMI requires a BMC to be present, the BMC is not required to have a Management
941 Controller Device Locator record in the Sensor Data Records. The BMC has a “Primary FRU
942 Device” which is accessed as logical FRU Device 0 on LUN 00b of the BMC using the *Read-*
943 and *Write FRU Data* IPMI commands.

944 **2.6.2 By Management Controller Device Locator Record**

945 Management Controllers other than the BMC are listed using individual Management Controller
946 Device Locator (Type records in the Sensor Data Records. A “Device Capabilities” field in the
947 record includes a bit that indicates whether or not the management controller supports a “Primary
948 FRU Inventory Device”. I.e. has FRU information that is accessible as logical FRU Device 0 on
949 LUN 00b of the management controller using the *Read-* and *Write FRU Data* IPMI commands.

950 **2.6.3 By FRU Device Locator Record**

951 FRU Devices that are not Primary FRU Devices are listed using individual FRU Device Locator
 952 records in the Sensor Data Records. The record provides information on whether the device is
 953 accessed as a Physical or Logical FRU Device.\

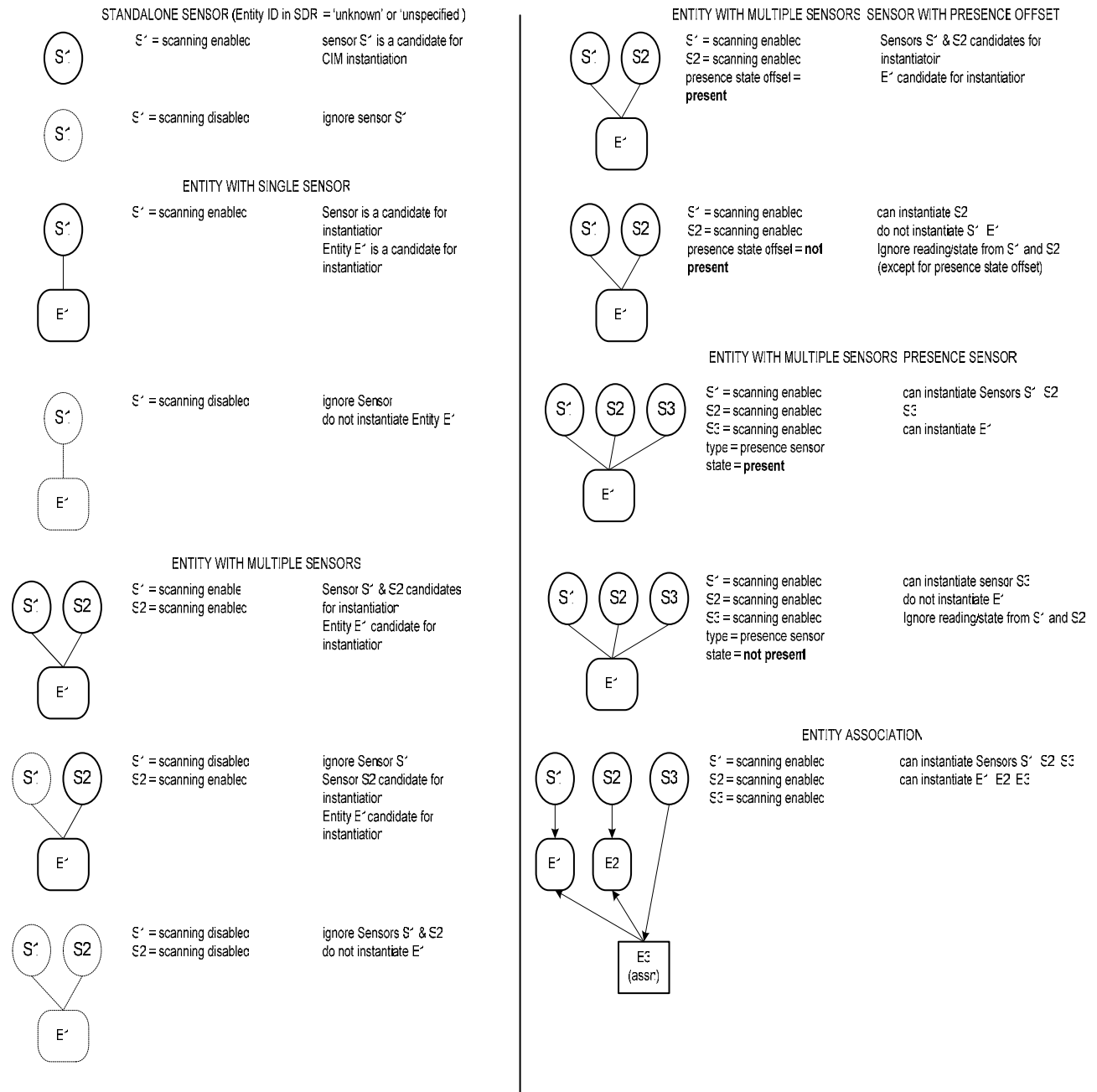
954 **2.7 Discovering Sensor, Entity, & FRU Associations**

955 The following figures graphically represent some common inter-relationships between sensors
 956 and Entities and how they relate to sensor and entity presence. In general, software uses the
 957 following information sources to determine and generate CIM associations between sensors,
 958 Entities, and FRU information:

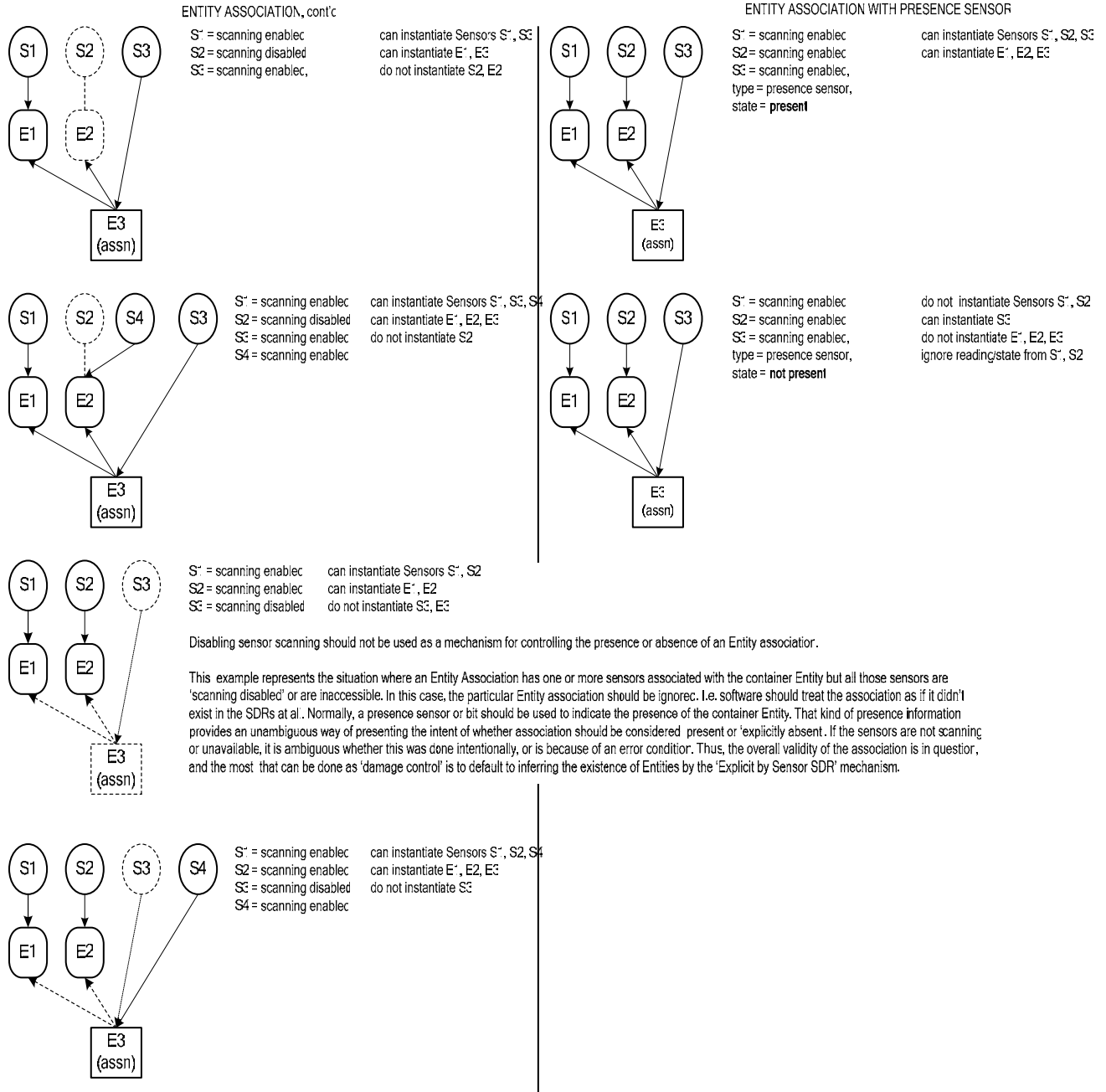
- 959 • **Entity ID Information in sensor SDRs:** Each sensor is associated with a single Entity,
 960 and there can be a multiple sensors associated with a given Entity. Software uses the
 961 Entity ID and Entity Instance information to identify the Entity that the sensor is
 962 associated with.
- 963 • **Entity Association records:** Entities can be associated with one another in a
 964 container/contained relationship via Entity Association records. Entity Association
 965 Records can be nested to form an “Entity Association Hierarchy”.
- 966 • **FRU- and Management Controller Device Locator records:** FRU Information is
 967 associated to Entities and management controllers via the Entity ID and Entity Instance
 968 fields in FRU Device Locator records and Management Controller Device Locator
 969 records.
- 970 • **BMC Primary FRU Device information:** When an Management Controller Device
 971 locator record is not present (as is typical for the BMC) the FRU information in the
 972 Primary FRU Device is assumed to be associated with the FRU that the BMC itself is
 973 associated with. For example, if there’s board information in the BMC Primary FRU, it
 974 is assumed that the information is for the board that the BMC is mounted on.
 975
- 976 • Chassis Info Area is FRU data associated with the overall chassis for the system.
 977 There should be only one set of Chassis Info Area data in the IPMI subsystem.
- 978 • Board Info Area, implies existence of a board or module or other FRU. The name
 979 ‘Board Info Area’ is somewhat a misnomer, because the usage is not restricted to
 980 just circuit boards. This area is also typically used to provide FRU information for
 981 any replaceable entities, boards, and/or sub-assemblies that are not sold as
 982 standalone products separately from other components. For example, individual
 983 boards from a board set, or a sub-chassis or backplane that’s part of a larger
 984 chassis. If an explicit Entity ID and Entity Instance is not associated with the FRU
 985 Device, it is assumed that the information is for the main system board.
- 986 • Product Info Area, if not associated with an explicit Entity via an Entity ID and
 987 Entity Instance, is assumed to be for the overall product. I.e. associated with the
 988 computer system.
 989

990 If any Entity has associated sensors, but they're all disabled or inaccessible, do not instantiate the
 991 Entity, period. If that Entity is the top container Entity of an Entity Association, apply the
 992 remaining Entity instantiation rules as if the particular Entity Association did not exist in the
 993 SDRs at all.

994 Otherwise, if there is at least one explicit sensor that indicates that the top entity of an Entity
 995 Association is present, use the Entity association, and instantiate or do not instantiate the
 996 container and contained Entities based on the presence of the association.



997
998



999
1000

1001 **2.8 Instantiating CIM Objects**

1002 **2.8.1 Sensor Instantiation :**

1003 After discovering the available IPMI sensors, the next step is to instantiate CIM Objects based on
 1004 the discovered sensors. This is done according to the Sensor Type information given in the SDR
 1005 for the sensor. Which particular CIM Object(s) should be instantiated for a given sensor is the
 1006 'mapping' process is described in more detail in Section 3, IPMI Sensor Mapping, and following
 1007 sections of this document.

1008 2.8.1.1 Sensor state/reading rules

1009 When populating attributes in CIM Objects that return state or reading information from a
1010 sensor, it is necessary to know when the IPMI sensor is returning valid information. This can
1011 generally be determined according to the following rules.

- 1012 4. ignore sensor state/reading if the *reading/state unavailable* bit for the sensor is set.
- 1013 5. if the sensor includes a 'presence' bit, ignore reading/state information for the sensor
1014 that requires the associated Entity to be present.
- 1015 6. if a sensor that includes a 'presence' bit is associated with the container Entity of an
1016 Entity-Association AND the sensor state = "not present", ignore the reading/state
1017 information for the sensor that requires the associated Entity to be present, and ignore
1018 the reading/state of any sensors for contained Entities in the Entity association.
- 1019 7. if an Entity Presence sensor is associated with the container Entity of an entity-
1020 association AND the sensor state = "not present", ignore the reading/state of any
1021 contained sensors in the Entity association.

1022 Note:

1023 Because IPMI specifies that disabled sensors (sensors that are set to 'sensor scanning disabled')
1024 should be ignored by software, it is recommended that instrumentation provider software does
1025 not disable sensors once they've been instantiated, since if the software was terminated, the
1026 sensors would remain disabled and a subsequent load of the software would ignore them.

1027 Instrumentation software should not assume that the initial disabled state of a sensor (e.g.
1028 scanning enabled or disabled) will remain the same across restarts of the operating system, since
1029 it's possible that a hardware change could have occurred between OS restarts that would cause
1030 the management controller to disable a sensor that had been previously enabled.

1031 Refer to Section 3.4.1.2, Method: RequestStateChange for additional information.

1032 2.8.2 Entity Instantiation

1033 Once the IPMI Entities have been discovered, a CIM Objects should be instantiated according to
1034 the Entity ID for the Entity according to the steps described in Section 10, IPMI Entity Mapping.
1035 In addition to instantiating CIM Objects based on the Entities, CIM associations are created that
1036 relate those objects to one another, and to a ComputerSystemPackage association under the host
1037 Computer System.

1038 2.8.3 FRU instantiation

1039 IPMI FRU information is used as part of the Entity Instantiation process described in Section 10,
1040 IPMI Entity Mapping. In general, the instrumentation provider software should apply discovered
1041 FRU information as follows:

- 1042 • Instantiate any FRU information associated with an instantiated Entity.

- 1043 • Instantiate any FRU information associated with the BMC and any present satellite
1044 management controllers.
- 1045 • Instantiate any FRU information for FRU devices listed in the SDRs and associated with an
1046 Entity that is otherwise unreferenced in the SDRs.
- 1047 • It is possible that FRU information could be available for an Entity that is instantiated but
1048 not present (this is because FRU information can be kept in separate devices from the
1049 Entity). Software should check for this possibility and should assume that if the Entity is not
1050 present, FRU Information for the Entity is not valid and should be ignored.

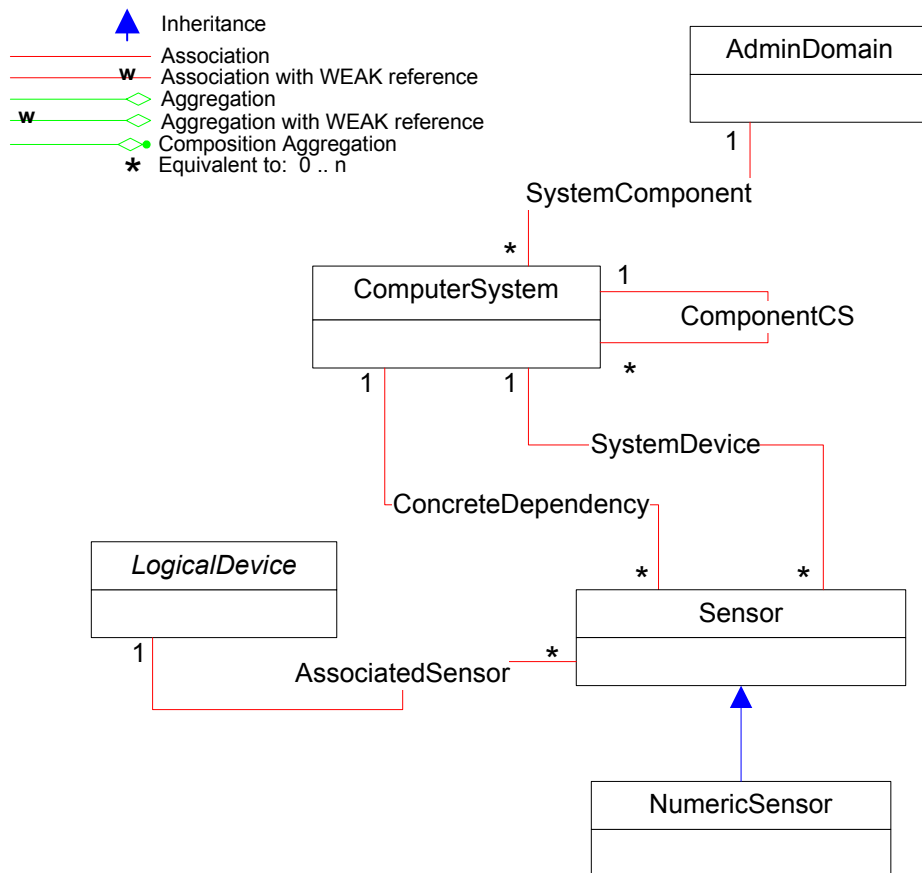
1051 3 IPMI Sensor Mapping

1052 3.1 Introduction

1053 IPMI Sensors can be mapped into the CIM model's CIM_Sensor class (and it's
 1054 CIM_NumericSensor specialization). Discrete sensors become instances of CIM_Sensors.
 1055 Analog sensors become instances of CIM_NumericSensor. IPMI entities, which are associated
 1056 with IPMI Sensors in the IPMI Sensor Data Record (SDR), typically become instances of
 1057 corresponding CIM_LogicalDevice's (or specializations thereof). The CIM_AssociatedSensor
 1058 connects each instance of a sensor to the corresponding IPMI-Entity/CIM_LogicalDevice.
 1059 Ultimately, all of these various class instances are associated with some enclosing instance of a
 1060 CIM_ComputerSystem.

1061 The following class diagram represents the various CIM classes that can be used when mapping
 1062 IPMI Sensors to the Common Information Model (CIM). The AdminDomain, ComputerSystem
 1063 and Logical Device classes are shown in this diagram to illustrate the way the Sensor classes are
 1064 associated with overall class ecosystem. This is in preparation for interoperability with
 1065 forthcoming (CIM) profiles from the DMTF and other standards bodies.

IPMI Sensor Class Diagram



1066

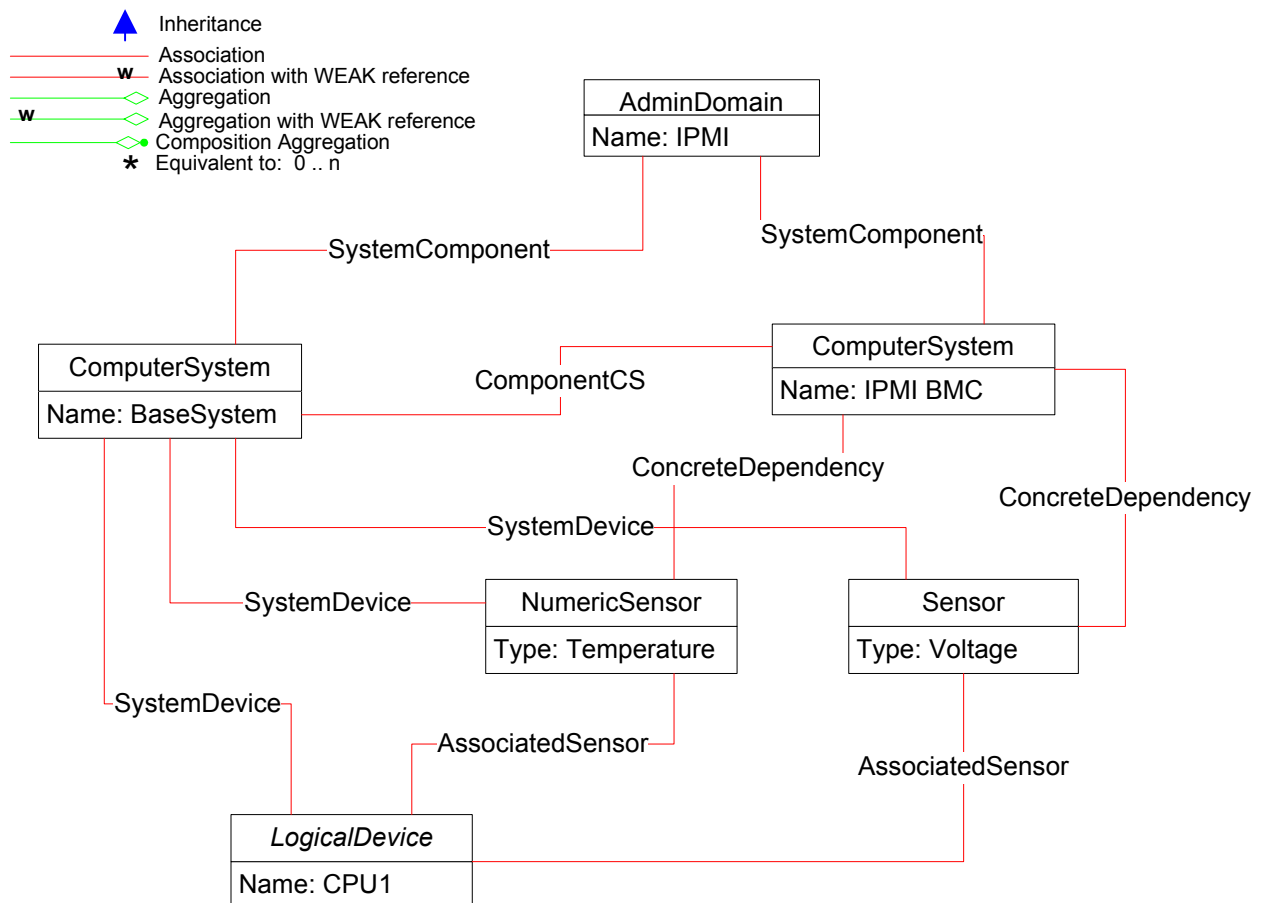
1067 **Figure 1 Class Diagram – IPMI Sensor Profile Mapping**

1068

1069 3.2 Instance Examples

1070 The following instance diagram represents that instantiation of a discrete voltage presence sensor
 1071 (ie voltage is good or bad) and an analog temperature sensor (returns value in a numeric range).
 1072 The AdminDomain, ComputerSystem and LogicalDevice instances are shown for illustration of
 1073 how the sensor instances are associated to an overall object model representation of a managed
 1074 system.

IPMI Sensor Instance Diagram



1075

1076

Figure 2 Instance Diagram – IPMI Sensor Profile Mapping

1077 3.3 Mapping Requirements

1078 3.3.1 Handling Sensor Specific Offsets

1079 IPMI sensors that use sensor specific offsets present a CIM mapping challenge. There are no
 1080 CIM classes that represent a sensor that itself represents multiple real-world sensors. Therefore,

1081 mapping IPMI sensors with Sensor-specific Offsets must involve the generation an instance of
1082 CIM_Sensor that asserts each Sensor-specific offset. Technically, this is necessary only for
1083 those Sensor-specific Offsets which are enabled. The IPMI SDR Readable Mask, Assert Mask,
1084 or Deassert Mask describe just which Sensor-specific Offsets are enabled.

1085 Instances of CIM_Sensor representing sensor specific offsets shall have a SensorType of “Other”
1086 and an OtherSensorTypeDescription set to a corresponding Event string value. The properly
1087 corresponding Event string values should be drawn, verbatim, from Table 36-3 of the IPMI 1.5
1088 Specification.

1089 The PossibleStates string array property shall be set to the following 2 strings:

- 1090 1. <Event string value from Table 36-3> || “Assert”
- 1091 2. <Event string value from Table 36-3> || “Deassert”

1092

1093 The CurrentState property shall have one of the 2 values of PossibleStates.

1094 **3.4 CIM_Sensor Mapping**

1095 **3.4.1 CIM_Sensor Methods**

1096 **3.4.1.1 Method: Reset()**

1097 `uint32 Reset()`

1098

1099 Per the DMTF definition: The return value should be 0 if the request was successfully executed,
1100 1 if the request is not supported and some other value if an error occurred. For the IPMI
1101 mapping this method would cause the Re-arm Sensor Events Command to be invoked for the
1102 sensor.

1103 Byte 2, bit 7 of the request data for the Rearm_Sensor_Events command should be set to 0 to
1104 rearm all events. Bytes 3-6 are not required to be sent.

1105 Return value:

1106 0 – Completed with no error

1107 1 – Not Supported

1108 2 – Unspecified Error

1109 3 – Timeout

1110 4 – Failed

1111 5 – Invalid Parameter

1112 6..0xFFF – DMTF_Reserved

1113 0x1000..0x7FFF – Method_Reserved

1114 0x8000.. – Vendor_Reserved

1115

1116 3.4.1.2 Method: RequestStateChange

```
1117 uint32 RequestStateChange (
1118     [IN] uint16 RequestedState,
1119     [IN] datetime TimeoutPeriod)
```

1120 **For the IPMI managed sensor, the current recommendation is that the method SHOULD**
 1121 **NOT be supported.** This recommendation is due to a sensor not being “discoverable” after
 1122 being disabled. The current definition for discovering a sensor requires that an SDR be
 1123 present for the sensor and the sensor enabled bit is on. If a sensor is dynamically disabled,
 1124 subsequent “discovery” of sensors by IPMI CIM mapping implementations will not show
 1125 the sensor.

1126 Return value:
 1127 1 – Not Supported

1128 3.4.2 CIM_Sensor Properties

1129 REQUIRED properties are properties that MUST contain values supplied via the IPMI sub-
 1130 system. RECOMMENDED properties are properties that MAY contain values supplied by the
 1131 IPMI subsystem.

1132 **Table 1 CIM_Sensor Class Properties**

Required Properties	Notes
SystemCreationClassName	Key:
SystemName	Key:
CreationClassName	Key:
DeviceID	Key:
SensorType	
OtherSensorTypeDescription	
Name	
CurrentState	
PossibleStates	
EnabledState	
HealthState	
OperationStatus	
<i>Recommended Properties</i>	Notes
PollingInterval	
StatusDescriptions	Not defined.
OtherIdentifyingInfo	Not defined.
IdentifyingDescriptions	Not defined.
AdditionalAvailability	Not defined.
InstallDate	Not defined.
RequestedState	Not defined.

1133

<i>Recommended Properties</i>	Notes
TimeOfLastStateChange	Not defined.
OtherEnabledState	Not defined.
RequestedState	Not defined.
EnabledState	Not defined.
Caption	
Description	
ElementName	

1134 **3.4.2.1 CIM_Sensor.SystemCreationClassName**

1135 Per the DMTF definition of the class property: The scoping System's CreationClassName. In all
1136 cases this will be the class "CIM_ComputerSystem".

1137 Formulate as a string:

1138 <"CIM_ComputerSystem">

1139 **3.4.2.2 CIM_Sensor.SystemName**

1140 Formulate as a string:

1141 "ComputerSystem.Name, of the ComputerSystem instance, across the SystemDevice
1142 association>

1143 The CIM_ComputerSystem instance supplying the CIM_ComputerSystemName property value
1144 is the CIM_ComputerSystem instance the 'supervises' (or 'owns') the given CIM_Sensor
1145 instance. Typically, this CIM_ComputerSystem instance represents the management controller
1146 associated with the given sensor. This string value must be found and then copied in to the
1147 CIM_Sensor.SystemName property. Since there is but one instance of a management controller
1148 for most IPMI-compliant devices/systems, this is often quite easy to do.

1149 Some modular systems may support an entire hierarchy of IPMI management controllers. For
1150 instance, the Advanced Telecommunications Architecture (ATCA – also known as PICMG 3.x),
1151 can support shelf management controllers (ShMCs). These are sometimes called Chassis
1152 Management Modules (CMMs – especially for bladed systems). Whatever the nomenclature,
1153 master management controllers often talk to some set of 'line' card or 'blade' card. Each such
1154 line/blade card can have its own embedded management controller. In turn, these (carrier) cards
1155 might support mezzanine cards (e.g. the ATCA mezzanine card standard). Once again, the
1156 mezzanine cards themselves might support IPMI management controllers (using names like the
1157 integrated platform management controller - IPMC). IPMI was designed to handle such
1158 hierarchical configurations of management controllers (and 'IPMI message bridging' amongst
1159 them).

1160 In the presence of a management controller hierarchy, the CIM_Sensor.SystemName value
1161 comes from the particular instance of a CIM_ComputerSystem (representing a management
1162 controller) that has a direct CIM_SystemDevice association with the sensor in question.

1163 3.4.2.3 CIM_Sensor.CreationClassName

1164 As per the DMTF definition of this class property: The name of the class or the subclass used in
 1165 the creation of an instance. This will typically be “CIM_Sensor” or “CIM_Numeric_Sensor”.
 1166 However, some CIM implementations may include one or more proprietary subclasses of
 1167 CIM_Sensor. In these cases the CIM_Sensor.CreationClassName value will be the name
 1168 identifying such a subclass.

1169 Formulate as a string:

1170 ”CIM_Sensor” or the name of the subclass created

1171 3.4.2.4 CIM_Sensor.DeviceID

1172 If SDR Record Type >= 01h and SDR Record Type <= 02h

1173 The DeviceID is generated from the concatenation of the following fields separated by
 1174 periods (".")

1175 SDR Type (1|2)

1176 Sensor Number (SDR type 1|2 table byte 8)

1177 Owner LUN (SDR type 1|2 table byte 7)

1178 Owner ID (SDR type 1|2 table byte 6)

1179 Event/Reading Type Code (hex to string)

1180 Sensor Specific Offset or "99" if none (hex to decimal to string)

1181 Formulate as a string:

1182 (string)(SensorNumber) || “.” || (string)(OwnerLUN) || “.” || (string)(OwnerID) ||

1183 (string) (Event/Reading Type Code> || ”.” ||

1184 (string) (Sensor Specific Offset or 99 if none)

1185 Examples:

1186 Discrete sensor, Full Record Typ 1, for Processor Sensor 6, LUN 0, Owner ID 1, Offset
 1187 07 (Presence Detected).

1188 1.6.0.1.6F.7

1189 Temperature sensor, Compact record Type 2, for Fan Sensor 8, LUN 0, Owner ID 2, no
 1190 offset

1191 2.8.0.2.01.99

1192 else if SDR Record Type == C0h

1193 The DeviceID is generated from the concatenation of the following fields separated by
 1194 periods (".")

1195 SDR Type (12)

1196 SDR Record ID(SDR table byte 1 and 2)

1197 Formulate as a string:

1198 (string) 12 || (string) (SDR Record ID(SDR Table byte 1 and 2))

1199 Example:

1200 OEM SDR with Record ID 0102h

1201 12.258

1202 **3.4.2.5 CIM_Sensor.ElementName and CIM_Sensor.Name**

1203 if SDR Record Type >= 01h and SDR Record Type <= 02h

1204 Formulate as a string:

1205 (string)<IPMI v1.5 SDR Record Type 01h or 02h, SDR entry bytes 49+, see IPMI v1.5
 1206 Table 37-1> || "(" || (string)(SensorNumber) || "." || (string)(OwnerLUN) || "." ||
 1207 string)(OwnerID) || ")"

1208 Examples:

1209 GeneralChassis1Intrusion(8.0.2)

1210 DriveBay3Intrusion(7.0.1)

1211 Processor4ThermalTrip(4.0.6)

1212 LAN2Leash(8.0.8)

1213 PUnit1OutputVoltage(6.0.1)

1214 CPU1T(13.1.3)

1215 else if SDR Record Type == C0h

1216 Formulate as a string:

1217 "OEM Record" || "(" || (string)(ManufacturerID) || ")"

1218 3.4.2.6 CIM_Sensor.SensorType and CIM_Sensor.OtherSensorTypeDescription

1219 When “present”, all non-discrete IPMI sensors will be mapped to a corresponding instance of a
 1220 NumericSensor. We say non-discrete, or analog, when we refer to sensors that provide a
 1221 variable reading value (like a number of volts or a number of degrees). We say discrete when
 1222 the sensor’s reading values map to a fixed, a-priori enumerated set of values. These enumeration
 1223 values are mapped by the IPMI specification to corresponding string values.

1224 More specifically, any IPMI sensor whose Event/Reading Type code is 01h (i.e. Event/Reading
 1225 Type code category == “threshold”) is a non-discrete/analog sensor. See the IPMI v1.5
 1226 specification, section 36.1 (especially tables 36-1 and 36-2). In both type 1 and type 2 SDRs, the
 1227 Event/Reading Type code is byte 14. See the IPMI specification’s table 37-1 and table 37-2 for
 1228 details.

1229 The discrete versus analog sensor type distinction is *not* based on the IPMI Sensor Type Code (as
 1230 per IPMI v1.5 specification Table 36-3). This might seem counter-intuitive, but this is the way
 1231 IPMI works. Even though some IPMI Sensor Types do call for a specific, enumerated set of
 1232 reading (offset) values, the Event/Reading Type code can override even these reserved values in
 1233 favor of other Event/Reading Type code specific offset values (see Table 36-2). While overrides
 1234 of this kind are extremely rare, they are possible. In the far more typical case, the Sensor (type)
 1235 specific (offset) values are used. Details on the Sensor-Specific Offset appear in Table 36-3)

1236 As mentioned, each analog sensor is mapped to a corresponding instance of a
 1237 CIM_NumericSensor if, and *only if*, that particular sensor is *present*. Consult the Sensor
 1238 Presence Analysis section above for the applicable presence detection algorithm (for any given
 1239 IPMI SDR described sensor).

1240 Formulate as a string:

1241 if <Sensor is present> and <Sensor is not disabled>

1242 if <Event/ReadingTypeCode> (>= 01h and <=0Ch) or == 6Fh

1243 if SensorTypeCode == 01h

1244 CIM_Sensor.SensorType = CIM Temperature

1245 else if SensorTypeCode == 02h

1246 CIM_Sensor.SensorType = CIM Voltage

1247 else if SensorTypeCode == 03h

1248 CIM_Sensor.SensorType = CIM Current (e.g. amps)

1249 else if SensorTypeCode == 04h

1250 CIM_Sensor.SensorType = CIM Tachometer

1251 else if SensorTypeCode == 05h

1252 CIM_Sensor.SensorType = CIM Other

1253 CIM_Sensor.OtherSensorTypeDescription = “ChassisIntrusion“

```
1254     else if SensorTypeCode == 06h
1255         CIM_Sensor.SensorType = CIM Other
1256         CIM_Sensor.OtherSensorTypeDescription = "PlatformSecurityViolation"
1257     else if SensorTypeCode == 07h
1258         CIM_Sensor.SensorType = CIM Other
1259         CIM_Sensor.OtherSensorTypeDescription = "Processor"
1260     else if SensorTypeCode == 08h
1261         CIM_Sensor.SensorType = CIM Other
1262         CIM_Sensor.OtherSensorTypeDescription = "PowerSupply"
1263     else if SensorTypeCode == 09h
1264         CIM_Sensor.SensorType = CIM Other
1265         CIM_Sensor.OtherSensorTypeDescription = "PowerUnit"
1266     else if SensorTypeCode == 0Ah
1267         CIM_Sensor.SensorType = CIM Other
1268         CIM_Sensor.OtherSensorTypeDescription = "CoolingDevice"
1269     else if SensorTypeCode == 0Bh
1270         CIM_Sensor.SensorType = CIM Other
1271         CIM_Sensor.OtherSensorTypeDescription = "UnitsBasedSensor"
1272     else if SensorTypeCode == 0Ch
1273         CIM_Sensor.SensorType = CIM Other
1274         CIM_Sensor.OtherSensorTypeDescription = "Memory"
1275     else if SensorTypeCode == 0Dh
1276         CIM_Sensor.SensorType = 1 # CIM Other
1277         CIM_Sensor.OtherSensorTypeDescription = "DriveSlotOrBay"
1278     else if SensorTypeCode == 0Eh
1279         CIM_Sensor.SensorType = 1 # CIM Other
```

```
1280     CIM_Sensor.OtherSensorTypeDescription = "POSTMemoryResize"
1281     else if SensorTypeCode == 0Fh
1282         CIM_Sensor.SensorType = CIM Other
1283         CIM_Sensor.OtherSensorTypeDescription = "SystemFirmwarePost"
1284     else if SensorTypeCode == 10h
1285         CIM_Sensor.SensorType = CIM Other
1286         CIM_Sensor.OtherSensorTypeDescription = "EventLoggingDisabled"
1287     else if SensorTypeCode == 11h
1288         CIM_Sensor.SensorType = CIM Other
1289         CIM_Sensor.OtherSensorTypeDescription = "Watchdog1"
1290     else if SensorTypeCode == 12h
1291         CIM_Sensor.SensorType = CIM Other
1292         CIM_Sensor.OtherSensorTypeDescription = "SystemEvent"
1293     else if SensorTypeCode == 13h
1294         CIM_Sensor.SensorType = CIM Other
1295         CIM_Sensor.OtherSensorTypeDescription = "CriticalInterrupt"
1296     else if SensorTypeCode == 14h
1297         CIM_Sensor.SensorType = CIM Other
1298         CIM_Sensor.OtherSensorTypeDescription = "Button"
1299     else if SensorTypeCode == 15h
1300         CIM_Sensor.SensorType = CIM Other
1301         CIM_Sensor.OtherSensorTypeDescription = "ModuleOrBoard"
1302     else if SensorTypeCode == 16h
1303         CIM_Sensor.SensorTypeCode = CIM Other
1304         CIM_Sensor.OtherSensorTypeDescription = "MicroControllerOrCoProcessor"
1305     else if SensorTypeCode == 17h
```

```
1306     CIM_Sensor.SensorTypeCode = CIM Other
1307     CIM_Sensor.OtherSensorTypeDescription = "AddInCard"
1308     else if SensorTypeCode == 18h
1309         CIM_Sensor.SensorTypeCode = CIM Other
1310         CIM_Sensor.OtherSensorTypeDescription = "Chassis"
1311     else if SensorTypeCode == 19h
1312         CIM_Sensor.SensorTypeCode = CIM Other
1313         CIM_Sensor.OtherSensorTypeDescription = "Chip Set"
1314     else if SensorTypeCode == 1Ah
1315         CIM_Sensor.SensorTypeCode = CIM Other
1316         CIM_Sensor.OtherSensorTypeDescription = "OtherFRU"
1317     else if SensorTypeCode == 1Bh
1318         CIM_Sensor.SensorTypeCode = CIM Other
1319         CIM_Sensor.OtherSensorTypeDescription = "CableOrInterconnect"
1320     else if SensorTypeCode == 1Ch
1321         CIM_Sensor.SensorTypeCode = CIM Other
1322         CIM_Sensor.OtherSensorTypeDescription = "Terminator"
1323     else if SensorTypeCode == 1Dh
1324         CIM_Sensor.SensorType = CIM Other
1325         CIM_Sensor.OtherSensorTypeDescription = "SystemBootInitiated"
1326     else if SensorTypeCode == 1Eh
1327         CIM_Sensor.SensorType = CIM Other
1328         CIM_Sensor.OtherSensorTypeDescription = "BootError"
1329     else if SensorTypeCode == 1Fh
1330         CIM_Sensor.SensorType = CIM Other
1331         CIM_Sensor.OtherSensorTypeDescription = "OSBoot"
```

```
1332     else if SensorTypeCode == 20h
1333         CIM_Sensor.SensorType = CIM Other
1334         CIM_Sensor.OtherSensorTypeDescription = "OSCriticalStop"
1335     else if SensorTypeCode == 21h
1336         CIM_Sensor.SensorType = CIM Other
1337         CIM_Sensor.OtherSensorTypeDescription = "Slot/Connector"
1338     else if SensorTypeCode == 22h
1339         CIM_Sensor.SensorType = CIM Other
1340         CIM_Sensor.OtherSensorTypeDescription = "SystemACPIPowerState"
1341     else if SensorTypeCode == 23h
1342         CIM_Sensor.SensorType = CIM Other
1343         CIM_Sensor.OtherSensorTypeDescription = "WatchDog2"
1344     else if SensorTypeCode == 24h
1345         CIM_Sensor.SensorType = CIM Other
1346         CIM_Sensor.OtherSensorTypeDescription = "PlatformAlert"
1347     else if SensorTypeCode == 25h
1348         CIM_Sensor.SensorType = CIM Other
1349         CIM_Sensor.OtherSensorTypeDescription = "EntityPresence"
1350     else if SensorTypeCode == 26h
1351         CIM_Sensor.SensorTypeCode = CIM Other
1352         CIM_Sensor.OtherSensorTypeDescription = "MonitorASIC-IC"
1353     else if SensorTypeCode >= C0h and SensorTypeCode <= FFh
1354         CIM_Sensor.SensorType = CIM Other
1355         CIM_Sensor.OtherSensorTypeDescription = "OEM"
1356     else
1357         # This sensor does not map to an instance of
```

```
1358     # NumericSensor. It also does not map to an
1359     # instance of a Sensor nor any specialization of
1360     # CIM_Sensor. This sensor is ignored in the IPMI-to-CIM
1361     # mapping.
1362 else # This sensor's Event/Reading Type Code != 01h
1363     # Thus, this sensor will become an instance of Sensor (see below)
1364 else # Either this sensor is not present or this sensor is disabled
1365     # Skip this sensor. It will not map to any CIM object instance.

1366 3.4.2.7 CIM_Sensor.PossibleStates
1367 Formulate as a string:
1368 if <Event/ReadingTypeCode> >= 01h and <Event/ReadingTypeCode> <= 0Ch
1369     # Get PossibleStates[] strings from Table 36-2. These Event/ReadingTypeCodes
1370     # use the "Generic States". See section 36.1 for details.
1371 else if <Event/ReadingTypeCode> == 6Fh # The Sensor Specific Event/Reading Type
1372     # As per the intent of section 3.3.1, Handling Sensor Specific Offsets
1373     for evs in EStr(<given the Sensor Type Code, get all corresponding Sensor-specific Offsets>)
1374         cur_sensor = #create new instance of CIM_Sensor
1375         curr_sensor.PossibleStates[0] = evs || " Assert" # The assert/deassert order is critical
1376                                     # to section 3.4.2.8
1377         curr_sensor.PossibleStates[1] = evs || " Deassert"
1378         See Section 3.3.1 Handling Sensor Specific Offsets
1379
1380 else if <Event/ReadingTypeCode> >= 70h and <Event/ReadingTypeCode> <= 7Fh
1381     # PossibleStates will be undefined, as discrete state information is specific to the OEM
1382     # identified by the Manufacturer ID for the IPM device that is providing access to the
1383     # sensor. Software needing Manufacturer ID can call the GetManufacturerID method to
1384     # obtain this information
1385 else
1386     # This is not an Event/ReadingTypeCode supported by the IPMI-to-CIM mapping.
```

1387 # Skip this sensor. It will not map to a CIM object instance.

1388 **3.4.2.8 CIM_Sensor.CurrentState**

1389 If the IPMI GetSensorReading command returns a byte 4, then map the bit

1390 # fields of byte 4 to one of the CIM_NumericSensor.PossibleStates above.

1391 # If not, then the CurrentState is “OK”.

1392 # If multiple bits are set in byte 4, See Section 3.3.1 Handling Sensor Specific Offsets.

```

1393 # Given the IPMI sensor identifier,
1394 # and the Get Sensor Reading Command byte 4/5 response,
1395 # reverse map to find the corresponding instances (plural) of CIM_Sensor.
1396 # Remember that a single IPMI sensor with Sensor Specific Offsets
1397 # can turn into a collection of CIM_Sensor instances – one for asserting
1398 # each sensor specific offset. I'll call the associated CIM_Sensor instances
1399 # related CIM Sensors.
1400 #
1401 States_Asserted[] = <byte 4 [7:0] from Get Sensor Reading Command &
1402                               Byte 5 [6:0] from Get Sensor Reading Command>
1403 Related_Sensors[] = sort-by-IPMI-offset<related_CIM_Sensors>
1404 i=0;
1405 for current_sensor in <Related_Sensors>
1406     if Sates_Asserted[i]
1407         current_sensor.CurrentState = current_sensor.PossibleState[0] # Assert
1408     else
1409         current_sensor.CurrentState = current_sensor.PossibleState[1] # Deassert
1410 
```

1411 **3.4.2.9 CIM_Sensor.Caption**

1412 if SDR Record Type >= 01h and SDR Record Type <= 02h

1413 Formulate as a string:

```

1414 (string)TruncAt(64, EStr(CIM_Sensor.SensorType)) || "(" || (string)(SensorNumber) || "."
1415 || (string)(OwnerLUN) || "." || (string)(OwnerID) || ")"

```

1416 Example:

1417 Voltage(6.0.1)

1418 Temperature(13.1.3)

1419 else if SDR Record Type == C0h

1420 Formulate as a string:

1421 “OEM Record” || ” (“ || (string)(ManufacturerID) || “)”

1422 **3.4.2.10 CIM_Sensor.Description**

1423 Formulate as a string:

1424 if SDR Record Type >= 01h and SDR Record Type <= 02h

1425 (string)<IPMI v1.5 SDR Record Type 01h or 02h, SDR entry bytes 49+, see IPMI v1.5
1426 Table 37-1> || “(” || (string)(SensorNumber) || “.” || (string)(OwnerLUN) || “.” ||
1427 string)(OwnerID) || “): ” ||(string)(EStr(<CIM_Sensor.SensorType>)) || “ for ” ||
1428 (string)(<IPMI_SDR_EntityID>) || “ “ || (string)(<IPMI_SDR_EntityInstanceNumber>)

1429 else if SDR Record Type == C0h

1430 (string) “OEM Record (“ || (string)(ManufacturerID) “) : SDR Type C0h - OEM Data” ||
1431 (uint8[56])(SDR entry bytes 9-64, see IPMI v1.5 Table 37.11)

1432 Example:

1433 PUnitVolt1(6.0.1): Voltage for Power Supply 1

1434 CPU1T(13.1.3): Temperature for processor 1

1435 OEM Record (343): SDR Type C0h - OEM Data (0Dh FBh 7Bh D0h 00h 00h 00h 00h
1436 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h
1437 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h
1438 00h 00h 00h)

1439 **3.4.2.11 CIM_Sensor.PollingInterval**

1440 Format as a string:

1441 The polling interval that the Sensor hardware or the instrumentation uses to determine the current
1442 state of the Sensor. Units are nanoseconds.

1443 **3.4.2.12 CIM_Sensor.HealthState**

1444 TBD. Currently being profiled in DMTF SMWG.

1445 **3.4.2.13 CIM_Sensor.OperationalStatus**

1446 TBD. Currently being profiled in DMTF SMWG.

1447 **3.5 CIM_NumericSensor Mapping**

1448 CIM_NumericSensor is subclassed from CIM_Sensor and inherits many properties. The
1449 properties in Table 2 CIM_NumericSensor Class Properties that are the same as for an instance
1450 of CIM_Sensor MUST be implemented as defined in the “See <property name>” reference in the
1451 Notes column for each property.

1452 **3.5.1 CIM_NumericSensor.Properties**

1453 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
 1454 properties that MAY be present. For example, there may be properties that are only filled out in
 1455 special configurations of the CIM Schema.

1456 Properties that listed in the Required section in Table 2 below MUST be implemented as defined
 1457 in the subsection following this table unless the Notes column for each property specifies the
 1458 defining reference.

1459 **Table 2 CIM_NumericSensor Class Properties**

Required Properties	Notes
SystemCreationClassName	Key: See CIM_Sensor.SystemCreationClassName mapping
SystemName	Key: See CIM_Sensor.SystemName Mapping
CreationClassName	Key: See CIM_Sensor.CreationClassName mapping
DeviceID	Key: See CIM_Sensor.DeviceID mapping
SensorType	See CIM_Sensor.SensorType mapping
OtherSensorTypeDescription	See CIM_Sensor.OtherSensorTypeDescription mapping
Name	See CIM_Sensor.SystemCreationClassName mapping
HealthState	See CIM_Sensor.HealthStatus mapping
OperationStatus	See CIM_Sensor.OperationStatus mapping
CurrentReading	
BaseUnits	
UnitModifier	
RateUnits	
<i>Recommended Properties</i>	Notes
NominalReading	
NormalMax	
NormalMin	
MaxReadable	
MinReadable	
Resolution	
Accuracy	
IsLinear	
Hysterisis	
SupportedThresholds	
EnableThresholds	
SettableThresholds	
PollingInterval	See CIM_Sensor.PollingInterval
PossibleStates	See CIM_Sensor.PossibleStates mapping

1460

<i>Recommended Properties</i>	Notes
CurrentState	See CIM_Sensor.CurrentState mapping
Caption	See CIM_Sensor.Caption mapping
Description	See CIM_Sensor.Description mapping
ElementName	See CIM_Sensor.ElementName mapping
StatusDescriptions	Not defined.
LowerThresholdNonCritical	
UpperThresholdNonCritical	
LowerThresholdCritical	
UpperThresholdCritical	
LowerThresholdFatal	
UpperThresholdFatal	
RequestedState	Not defined.
EnabledState	Not defined.
TimeOfLastStateChange	Not defined.
OtherEnabledState	Not defined.

1461

1462 **3.5.1.1 NumericSensor.CurrentReading**

1463 The current value indicated by the Numeric Sensor.

1464 **3.5.1.2 NumericSensor.UnitModifier, BaseUnits, And RateUnits**

1465 We must do more than just map IPMI sensor's with an obviously matching IPMI Sensor Unit
1466 Type Code. An obvious match would be IPMI 'seconds' matching to CIM_Sensor.BaseUnits ==
1467 'seconds'. Scaling issues are less obvious. For example, IPMI 'milliseconds' must map to
1468 CIM_Sensor.BaseUnits = 'seconds' with a corresponding CIM_Sensor.UnitModifier = -3. The
1469 scaling required might sometimes be a bit less trivial. Still, all IPMI "BaseUnits" must be
1470 mapped to corresponding CIM "BaseUnits".

1471 CIM represents floating point values as signed integers (sint32) that are scaled. IPMI tends to
1472 use floating point numeric representations (e.g. IEEE 754/854). To map these into signed
1473 integers, we'll just assume a default "precision" value of "-2" (for the CIM member
1474 "UnitModifier"). We'll round the IPMI floating point value appropriately and then convert it to
1475 integer required. Recall that a CIM UnitModifier of -2 means that the CIM CurrentReading will
1476 be will multiplied by 0.01 (10^{-2}). So a CurrentReading of 333, where the CIM BaseUnits is
1477 "Voltage", means that the true current reading is 3.33v.

1478 The RateUnits from the IPMI SDR match one-to-one with the enumerated
1479 CIM_Sensor.RateUnits. The last three CIM_Sensor.RateUnits are not used by IPMI (e.g. "per
1480 week", "per month", "per year").

1481 Examples: BaseUnits

1482 EStr(5) == "Volts"

1483 EStr(3) == “Degrees F”

1484 Examples: RateUnits

1485 EStr(0) == “None”

1486 EStr(1) == “Per Microsecond”

1487 EStr(2) == “Per Millisecond

1488 EStr(3) == “Per Second”

1489 EStr(4) == “Per Minute”

1490 EStr(5) == “Per Hour”

1491 EStr(6) == “Per Day”

1492 **3.5.1.3 NumericSensor.NominalReading**

1493 NominalReading indicates the 'normal' or expected value for the NumericSensor.

1494 **3.5.1.4 NumericSensor.NormalMax**

1495 NormalMax provides guidance for the user as to the normal maximum range for the
1496 NumericSensor

1497 **3.5.1.5 NumericSensor.NormalMin**

1498 NormalMin provides guidance for the user as to the normal minimum range for the
1499 NumericSensor.

1500 **3.5.1.6 NumericSensor.MaxReadable**

1501 MaxReadable indicates the largest value of the measured property that can be read by the
1502 NumericSensor.

1503 **3.5.1.7 NumericSensor.MinReadable**

1504 MinReadable indicates the smallest value of the measured property that can be read by the
1505 NumericSensor.

1506 **3.5.1.8 NumericSensor.Resolution**

1507 Resolution indicates the ability of the Sensor to resolve differences in the measured property.
1508 The units for this measurement are determined by "BaseUnit*UnitModifier/RateUnit.

1509 3.5.1.9 NumericSensor.Accuracy

1510 Indicates the accuracy of the Sensor for the measured property. Its value is recorded as
1511 plus/minus hundredths of a percent. Accuracy, along with Resolution, is used to calculate the
1512 actual value of the measured physical property. Accuracy may vary depending on whether the
1513 Device is linear over its dynamic range.

1514 Example value: 0

1515 3.5.1.10 NumericSensor.IsLinear

1516 Indicates that the Sensor is linear over its dynamic range.

1517 Example value: TRUE

1518 3.5.1.11 NumericSensor.Hysteresis

1519 Indicates the margin built around the thresholds. This margin prevents unnecessary state changes
1520 when the Sensor reading may fluctuate very close to its thresholds. This could be due to the
1521 Sensor's tolerance/accuracy/resolution or due to environmental factors. Once a threshold is
1522 crossed, the state of the Sensor should change. However, the state should not fluctuate between
1523 the old and new states unless the Sensor's change in the reading exceeds the hysteresis value. The
1524 units for this measurement are determined by $\text{BaseUnit} * \text{UnitModifier} / \text{RateUnit}$.

1525 Fixed value of: 0

1526 3.5.1.12 NumericSensor.SupportedThresholds

1527 If supported by BMC (from SDR).

1528 3.5.1.13 NumericSensor.EnableThresholds

1529 If supported by BMC (from SDR).

1530 3.5.1.14 NumericSensor.SettableThresholds

1531 If supported by BMC (from SDR).

1532 3.5.1.15 NumericSensor.LowerThresholdNonCritical

1533 The Sensor's threshold values specify the ranges (min and max values) for determining whether
1534 the Sensor is operating under Normal, NonCritical, Critical or Fatal conditions. If Current
1535 Reading is between LowerThresholdNonCritical and Upper ThresholdNonCritical, then the
1536 Sensor is reporting a normal value. If CurrentReading is between LowerThresholdNonCritical
1537 and LowerThresholdCritical, then the CurrentState is NonCritical.

1538 Example value: 50

1539 3.5.1.16 NumericSensor.UpperThresholdNonCritical

1540 The Sensor's threshold values specify the ranges (min and max values) for determining whether
1541 the Sensor is operating under Normal, NonCritical, Critical or Fatal conditions. If the
1542 CurrentReading is between LowerThresholdNonCritical and UpperThresholdNonCritical, then
1543 the Sensor is reporting a normal value. If the CurrentReading is between UpperThreshold
1544 NonCritical and UpperThresholdCritical, then the CurrentState is NonCritical.

1545 Example value: 500

1546 3.5.1.17 NumericSensor.LowerThresholdCritical

1547 The Sensor's threshold values specify the ranges (min and max values) for determining whether
1548 the Sensor is operating under Normal, NonCritical, Critical or Fatal conditions. If the
1549 CurrentReading is between LowerThresholdCritical and Lower ThresholdFatal, then the
1550 CurrentState is Critical.

1551 Example value: -1200

1552 3.5.1.18 NumericSensor.UpperThresholdCritical

1553 The Sensor's threshold values specify the ranges (min and max values) for determining whether
1554 the Sensor is operating under Normal, NonCritical, Critical or Fatal conditions. If the
1555 CurrentReading is between UpperThresholdCritical and Upper ThresholdFatal, then the
1556 CurrentState is Critical.

1557 Example value: 550

1558 3.5.1.19 NumericSensor.LowerThresholdFatal

1559 The Sensor's threshold values specify the ranges (min and max values) for determining whether
1560 the Sensor is operating under Normal, NonCritical, Critical or Fatal conditions. If the
1561 CurrentReading is below LowerThresholdFatal, then the Current State is Fatal.

1562 Example value: -1279

1563 3.5.1.20 NumericSensor.UpperThresholdFatal

1564 The Sensor's threshold values specify the ranges (min and max values) for determining whether
1565 the Sensor is operating under Normal, NonCritical, Critical or Fatal conditions. If the
1566 CurrentReading is above UpperThresholdFatal, then the Current State is Fatal.

1567 Example value: 1270

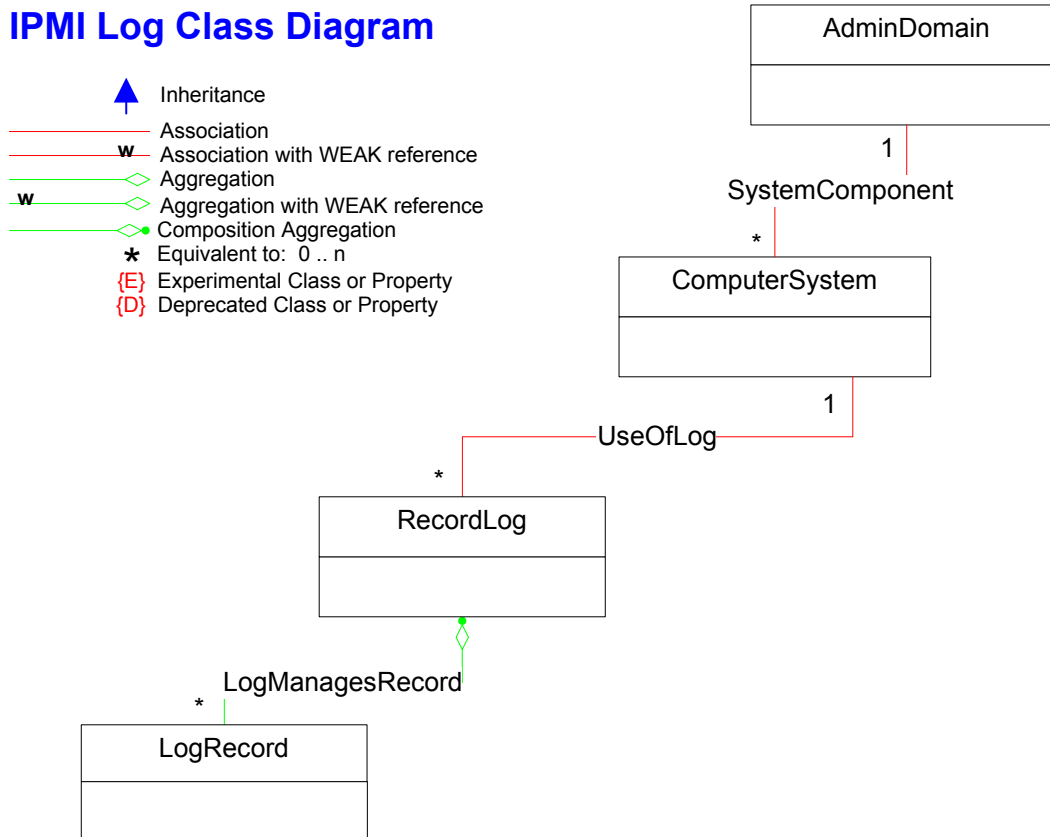
1568

1569 4 IPMI Log Mapping

1570 4.1 Introduction

1571 The IPMI Log Mapping extends the management capability of the IPMI to CIM mapping by
 1572 adding the capability to access and manage the IPMI System Event Log (SEL) entries via the
 1573 CIM_RecordLog and CIM_LogRecord classes.

1574 The following class diagram describes the CIM classes involved in representing IPMI System
 1575 Event Log by the CIM_RecordLog class and an individual SEL entries with the CIM LogRecord
 1576 class. The AdminDomain and ComputerSystem classes are shown in this diagram to illustrate
 1577 the way the RecordLog class is associated with overall class ecosystems. This is in preparation
 1578 for interoperability with forthcoming profiles from the DMTF and other standards bodies.



1579

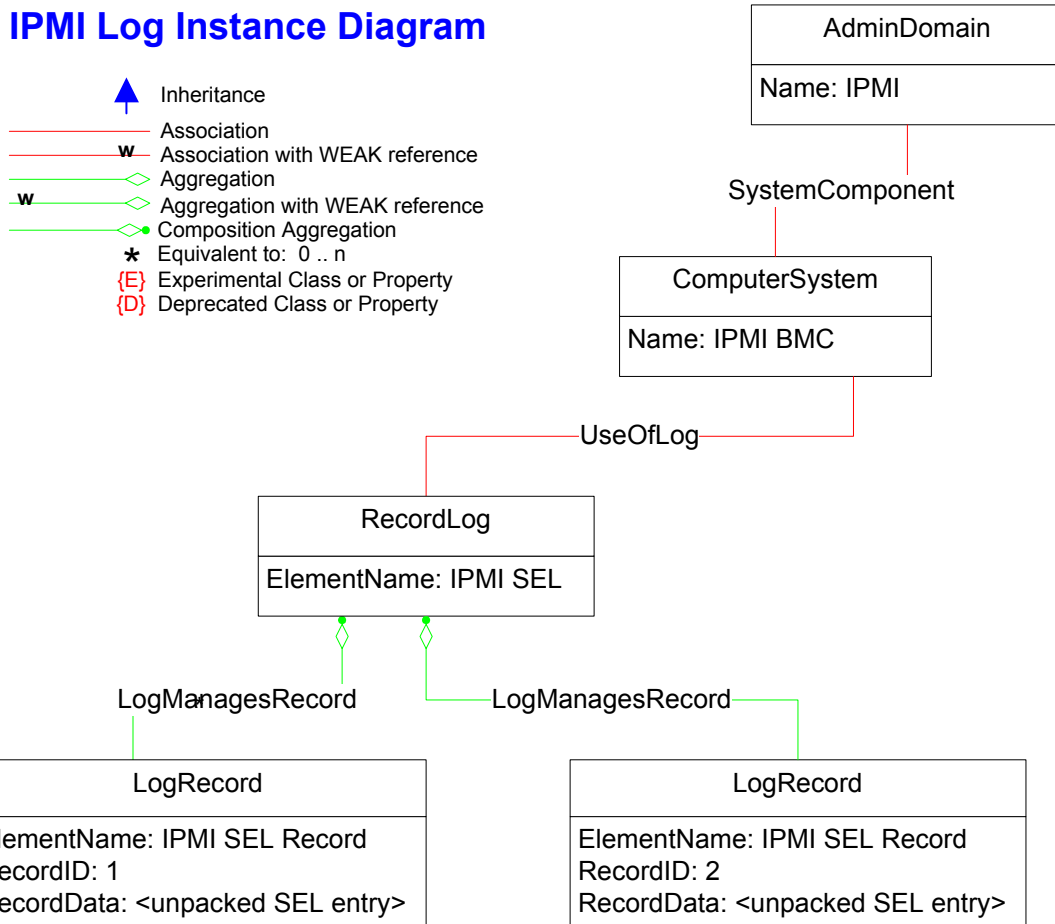
1580 **Figure 3 Class Diagram - IPMI Log**

1581 Note: The CIM_UseOfLog and CIM_LogManagesRecord associations are changed from
 1582 previous versions of the IPMI CIM Mapping Guideline due to changes in the DMTF Record Log
 1583 Profile.

1584 **4.2 Instance Diagrams**

1585 The IPMI Log Mapping is the CIM object model to IPMI interface mapping needed to manage
 1586 the IPMI SEL (System Event Log) exposed by an embedded Baseboard Management Controller
 1587 or Service Processor with IPMI compliant firmware. The objects modeled in this profile are
 1588 related to the IPMI BMC SEL and SEL entries specific to the IPMI Specification.

1589 The following instance diagram represents that instantiation of 2 SEL entries as LogRecord
 1590 instances and the SEL log by a RecordLog instance. The RecordLog instance has the methods
 1591 used to manipulate the SEL (ex. ClearLog). The AdminDomain and ComputerSystem instances
 1592 are shown for illustration of how the SEL log and entries instances are associated to an overall
 1593 object model representation of a managed system.



1594

1595 **Figure 4 Instance Diagram – IPMI Log**

1596 **4.3 Mapping Requirements**

1597 **4.3.1 Associating SEL with SDR**

1598 A SEL record’s associated CIM_Sensor instance (whether that is a NumericSensor or a Sensor
 1599 instance) can be found using the SEL Record’s GeneratorID (bytes 8-9) and Sensor Number

1600 (byte 12). See Table 26-1 for details. This SEL data maps to the CIM_Sensor.DeviceID key
1601 value (as defined in the IPMI-to-CIM sensor mapping document). Once this Sensor key string is
1602 extracted from the SEL entry in question, the corresponding CIM_Sensor instance can be found.

1603 Behind the scenes, the corresponding CIM_Sensor maps to a corresponding IPMI SDR record
1604 (or either type 1 or type 2). This indirectly corresponding SDR record must also be found in
1605 order to extract the IPMI_SDR_EntityID and IPMI_SDR_EntityInstanceNumber mentioned
1606 above.

1607 We refer to the CIM_Sensor instance above because CIM_Sensor is a common base class for
1608 both NumericSensor and Sensor (as “trivially derived” for this IPMI-to-CIM mapping).

1609 **4.4 CIM_RecordLog Mapping**

1610 **4.4.1 CIM_RecordLog.Methods**

1611 **4.4.1.1 Method: ClearLog**

1612 `uint32 ClearLog()`

1613 MUST delete all SEL records:

1614 An application clearing the SEL should utilize the recommended algorithm described below:

- 1615 a) first check to see if SEL is empty by checking the number of records using "Get SEL
1616 Info"
- 1617 b) if already empty, you're done
- 1618 c) if SEL not empty, use "Clear SEL" to see if erase is already in progress (in case Get
1619 SEL Info doesn't return correct status)
- 1620 d) if erase already in progress then wait (and potentially time out) and exit
- 1621 e) otherwise (if SEL is not empty) issue the Clear SEL command, then wait (and
1622 potentially time out) and exit

1623 Note: Originally, the ClearLog method on the MessageLog class would issue an IPMI
1624 request to clear the log and return success. But the log may not have actually have cleared
1625 by the time success is returned. The change made was to issue the clear request and spin for
1626 sometime checking if the operation was still in progress. After some period, the operation
1627 will timeout and a timeout error is returned to the client. This means the client needs to call
1628 ClearLog again (and also spin). This scenario is addressed by having the application utilize
1629 the algorithm described above.

1630 Return value:

- 1631 0 – Completed with no error
- 1632 1 – Not Supported
- 1633 2 – Unspecified Error
- 1634 3 – Timeout
- 1635 4 – Failed
- 1636 5 – Invalid Parameter

1637 6..0x0FFF – DMTF_Reserved
 1638 0x1000..0x7FFF – Method_Reserved
 1639 0x8000.. – Vendor_Reserved

1640 4.4.1.2 Method: RequestStateChange

```
1641 uint32 RequestStateChange (
1642     [IN] uint16 RequestedState,
1643     [IN] datetime TimeoutPeriod)
```

1644 For the IPMI SEL log, the action this method takes on the log is to freeze or stop the log if
 1645 the log from accepting further input if the RequestedState parameter is “Disabled”. If the
 1646 RequestedState parameter is “Enabled”, the IPMI SEL log is unfrozen or will continue to
 1647 accept log entries in the normal fashion. Not all IPMI implementations support the freezing
 1648 of the IPMI SEL log and these implementation will need to return “Not Supported” if the
 1649 RequestStateChange method is invoked.

1650 Return value:

1651 0 – Completed with no error
 1652 1 – Not Supported
 1653 2 – Unspecified Error
 1654 3 – Timeout
 1655 4 – Failed
 1656 5 – Invalid Parameter
 1657 6..0x0FFF – DMTF_Reserved
 1658 0x1000..0x7FFF – Method_Reserved
 1659 0x8000.. – Vendor_Reserved

1660 4.4.2 CIM_RecordLog.Properties

1661 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
 1662 properties that MAY be present. For example, there may be properties that are only filled out in
 1663 special configurations of the CIM Schema.

1664 Properties that listed in the Required section in Table 2 below MUST be implemented as defined
 1665 in the subsections following this table unless the Notes column for each property specifies the
 1666 defining reference.

1667 **Table 3 CIM_RecordLog Class Properties**

Required Properties	Notes
InstanceID	Key:
Name	
MaxNumberOfRecords	
CurrentNumberOfRecords	
EnabledState	
OperationalStatus	
HealthState	

1668

<i>Recommended Properties</i>	<i>Notes</i>
OtherEnabledState	Not defined.
RequestedState	Not defined.
EnabledDefault	Not defined.
TimeOfLastStateChange	Not defined.
InstallDate	Not defined.
StatusDescriptions	Not defined.
Caption	
Description	
ElementName	

1669

1670 The following subsections describe procedures for generating values for various properties of
1671 CIM classes required in this profile.

1672 **4.4.2.1 CIM_RecordLog.InstanceID**

1673 Per the DMTF definition: In order to ensure uniqueness within the NameSpace, the value of
1674 InstanceID SHOULD be constructed using the following ' preferred ' algorithm: < OrgID > : <
1675 LocalID > Where < OrgID > and < LocalID > are separated by a colon ' : '.

1676 Formulate as a string:

1677 <"IPMI" || ":" || < ComputerSystem.Name of the ComputerSystem instance of the one
1678 associated BMC > || " SEL Log">

1679 If there are several BMC's in the same namespace, unique names for the BMC's need to be
1680 generated in order to distinguish between the different instantiations of RecordLog. The name of
1681 the BMC can be formulated by instrumentation using unique identifying values obtained from
1682 the BMC or derived from instrumentation access deviations.

1683 **4.4.2.2 CIM_RecordLog.Name**

1684 Fixed value of "IPMI SEL"

1685 **4.4.2.3 CIM_RecordLog.MaxNumberOfRecords**

1686 Calculated by instrumentation using the following approach:

1687 The maximum number of SEL records can be computed by issuing the Get_Sel_Allocation
1688 command and using the response data. Dividing the number of free allocation units (bytes 4,5)
1689 by the maximum record size in allocation units (byte 10) will provide the maximum number of
1690 records. In the event that Get_Sel_Allocation command is not supported, the value should be set
1691 to 0.

1692 **4.4.2.4 CIM_RecordLog.CurrentNumberOfRecords**

1693 Calculated by instrumentation using the following approach:

1694 The current number of log records can be determined by issuing a Get_Sel_Info command and
1695 examining the number of log entries (bytes 3,4) in the response data

1696 **4.4.2.5 CIM_RecordLog.EnabledState**

1697 Per definition in CIM mof: EnabledState is an integer enumeration that indicates the
1698 enabled/disabled states of an element. For the IPMI mapping, the value for EnabledState can
1699 only be “Enabled” and “Disabled”. Enabled indicates the SEL is active and accepting new
1700 entries. Disabled indicates the log is “frozen” or full. Frozen is when the SEL is
1701 programmatically suspended to not accept further entries and is available in some IPMI
1702 implementations (See the RequestStateChange method definition).

1703 Formulate as an integer:

1704 2 – Enabled

1705 3 - Disabled

1706 **4.4.2.6 CIM_RecordLog.HealthState**

1707 TBD. Currently being profiled in DMTF SMWG.

1708 **4.4.2.7 CIM_RecordLog.OperationalStatus**

1709 TBD. Currently being profiled in DMTF SMWG.

1710 **4.4.2.8 CIM_RecordLog.Caption**

1711 Fixed value of “IPMI SEL”

1712 **4.4.2.9 CIM_RecordLog.Description**

1713 Fixed value of “IPMI SEL”

1714 **4.4.2.10 CIM_RecordLog.ElementName**

1715 Fixed value of “IPMI SEL”

1716 **4.5 CIM_LogRecord Mapping**

1717 **4.5.1 CIM_LogRecord Methods**

1718 None Defined.

1719 **4.5.2 CIM_LogRecord Properties**

1720 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
 1721 properties that MAY be present. For example, there may be properties that are only filled out in
 1722 special configurations of the CIM Schema.

1723 Properties that listed in the Required section in Table 2 below MUST be implemented as defined
 1724 in the subsection following this table unless the Notes column for each property specifies the
 1725 defining reference.

1726 **Table 4 CIM_LogRecord Class Properties**

Required Properties	Notes
LogCreationClassName	Key:
LogName	Key:
CreationClassName	Key:
RecordID	Key:
MessageTimestamp	Key:
RecordFormat	
RecordData	
<i>Recommended Properties</i>	Notes
Caption	
Description	
ElementName	

1728 **4.5.2.1 CIM_LogRecord.LogCreationClassName**

1729 Per the DMTF definition of the class property: The name of the class or the subclass used in the
 1730 creation of an instance of the log for with this record is a part. This will typically be
 1731 “CIM_RecordLog”.

1732 Formulate as a string:

1733 <“CIM_RecordLog” or the name of the subclass created for the containing log>

1734 **4.5.2.2 CIM_LogRecord.LogName**

1735 Propagated ("CIM_RecordLog.Name")

1736 Fixed value of: “IPMI SEL”

1737 **4.5.2.3 CIM_LogRecord.CreationClassName**

1738 Per the DMTF definition of the class property: The name of the class or the subclass used in the
 1739 creation of an instance. This will typically be “CIM_LogRecord”.

1740 Formulate as a string:

1741 <"CIM_LogRecord">

1742 **4.5.2.4 CIM_LogRecord.RecordID**

1743 RecordID is a Key and must not be Null. The value will be populated with the IPMI SEL
1744 Record ID. Used in conjunction with the MessageTimestamp, must uniquely identify an
1745 instance of LogRecord that is associated via RecordInLog to the containing instance of
1746 RecordLog representing the SEL.

1747 Formulate as a string:

1748 <SEL Record ID as passed by the Get_SEL_Entry IPMI command>

1749 **4.5.2.5 CIM_LogRecord.MessageTimestamp**

1750 This value is the timestamp of the SEL entry plus an offset to GMT. There is an assumption that
1751 the BMC time will be set by the system management software running "in-band" in the managed
1752 system that the BMC resides. In band instrumentation will generate the GMT offset required by
1753 this data type by discerning the timezone that the managed system is in and producing the
1754 necessary offset. Out-of-band instrumentation doing CIM mapping will need to determine the
1755 difference with the BMC time and add that difference to the offset for the timezone the out-of-
1756 band client is in.

1757 Formulate as a string using the datetime data type as defined by the DMTF:

1758 Example value: "20040924152230.000000+480"

1759 For non-timestamped OEM SEL, this property is set to "Unknown" as defined by the DMTF:

1760 Value = "99990101000000.000000+000"

1761 This definition requires that the BMC time **MUST** be set by in-band instrumentation to the time
1762 of the owning managed system. This is typically done today and is a Best Practice.

1763 **4.5.2.6 CIM_LogRecord.RecordFormat**

1764 A free-form string describing the LogRecord's data structure. To describe the data structure of
1765 RecordData, the RecordFormat string should be constructed as follows: The first character is a
1766 delimiter character and is used to parse the remainder of the string into sub-strings. Each sub-
1767 string is separated by the delimiter character and should be in the form of a CIM property
1768 declaration (i.e., datatype and property name). This set of declarations may be used to interpret
1769 the similarly delimited RecordData property

1770 For system events, value of:

1771 "**string IPMI_SensorNumber.IPMI_OwnerLUN.IPMI_OwnerID*uint8[2]
1772 IPMI_RecordID*uint8 IPMI_RecordType*uint8[4] IPMI_Timestamp*uint8[2]
1773 IPMI_GeneratorID*uint8 IPMI_EvMRev*uint8 IPMI_SensorType*uint8

1774 IPMI_SensorNumber*boolean IPMI_AssertionEvent*uint8 IPMI_EventType*uint8
1775 IPMI_EventData1*uint8 IPMI_EventData2*uint8 IPMI_EventData3*uint32 IANA*"

1776 For timestamped OEM events, value of:
1777 "*uint8[2] IPMI_RecordID*uint8 IPMI_RecordType*uint8[4] IPMI_Timestamp*uint8[3]
1778 IPMI_ManufacturerID*uint8[6] IPMI_OEMDefinedData*uint32 IANA*"

1779 For non-timestamped OEM events, value of:
1780 "*uint8[2] IPMI_RecordID*uint8 IPMI_RecordType*uint8[13] IPMI_OEMDefinedData*"

1781 If the SEL timestamp has a value of 0xFFFFFFFF (which indicates an invalid or unspecified
1782 time value) or 0x00000000 through 0x20000000 (which indicates a time value that is relative
1783 BMC power-up and NOT 1970), a value of 0 should be set for CIM_LogRecord.
1784 MessageTimestamp.

1785 **4.5.2.7 CIM_LogRecord.RecordData**

1786 The following are examples of the Record.Data value.

1787 System events:
1788 for normal events, values like:
1789 ""10.0.32*16 0*2*182 59 84 65*32 0*4*1*10*true*1*87*99*90*1*"
1790 for shutdown events, values like:
1791 ""0.0.65*48 0*2*182 99 84 65*65 0*4*32*0*true*111*3*255*255*1*"
1792 for BSOD events, values like:
1793 ""0.0.65*96 0*2*182 5 85 65*65 0*4*32*0*true*111*2*255*255*1*"

1794 Timestamped OEM events, values like:
1795 ""128 0*192*56 87 84 65*94 43 0*4 128 0 0 0 0*1*"

1796 Non-Timestamped OEM events, values like:
1797 ""256 0*253*56 87 84 65 0 1 0 4 128 0 0 0 0*"

1798 While ordinary users of the CIM interface may not understand the specific data encoded into a
1799 RecordData property, a few users and OEM instrumentation and IPMI aware applications might
1800 know enough about IPMI to find this data useful. This is what the CIM model intends for this
1801 deliberately free-form (RecordData) property. See the IPMI specification, table 26-1, bytes 8-16
1802 for details on what we expose in RecordData.

1803 Aside from IPMI details, we also expose the CIM_Sensor.DeviceID key. This key value
1804 uniquely identifies the instance of CIM_Sensor (i.e. either NumericSensor or Sensor) that is
1805 associated with this particular SEL record.

1806 **4.5.2.8 CIM_LogRecord.Caption**

1807 EStr(<SEL Sensor.Type>) || EStr(<SEL Generic or Sensor-Specific State Code>)

1808 *Procedure* for EStr(<SEL Generic or Sensor Specific State Code>)

1809 # SEL RecordType is byte 3 of SEL record (see Table 26-1)
1810 # Only SEL RecordType == 02h is a “system event record”.
1811 # Others are OEM defined, reserved, etc. All further comments
1812 # tend to assume SEL RecordType == 02h.
1813
1814 # SEL Event Message Revision (EvM) is byte 10 of SEL Record
1815 # (see Table 26-1). IPMI v1.5 uses 04h.
1816 # *Future Direction:* Support IMPI v1.0 (03h) and/or IPMI v2.0 (05h?)
1817
1818 # SEL Event Class is byte 13, bits [6:0] of SEL Record (see Table 26-1).
1819 # The three possible values for EventClass are taken from
1820 # Event/ReadingReadingTypeCodes as defined in Table 36-1.
1821 # The three possible EventClass values are listed in Table 23-6.
1822 # The EventClass dictates how to interpret the 3 subsequent bytes of
1823 # Event Data found in each SEL Record’s bytes 14-16. We call
1824 # these three Event Data bytes EB1, EB2 and EB3.
1825
1826 # SEL Event Direction is byte 13, bit [7] of each SEL
1827 # Record. We abbreviate the each SEL Record’s Event
1828 # Direction as EDir.
1829
1830 # SELSensorType is byte 11 of the SEL Record (see Table 26-1).
1831
1832 if <SEL RecordType> == 02h and <EvM> == 04h
1833 if <SEL Event Class > == 01h or
1834 (<SEL Event Class > >= 02h and <SEL Event Class> <= 0Ch)

```
1835     if <SEL EDir> == 1b
1836         strEDir = "Assert: "
1837     else # <SEL EDir> == 0b
1838         strEDir = "Deassert: "
1839     if <SEL EB1 [7:6]> == 11b
1840         # Use SEL SensorType and EB2 value to index into Table 36-3
1841         # to find a Sensor Specific Offset/State description string. EB2
1842         # is the Sensor-Specific Offset/State value..
1843         strOffsetOrStateDesc = strEDir + <EStr(<EB2 into Table36-3>)>
1844         return strOffsetOrStateDesc
1845     else if <SEL EB1 [6:5]> == 11b
1846         # Use SEL SensorType and EB3 to index into Table 36-3 to find
1847         # a Sensor Specific Offset/State description string. EB3
1848         # is the Sensor-Specific Offset/State value.
1849         strOffsetOrStateDesc = strEDir + <EStr(<EB3 into Table36-3>)>
1850         return strOffsetOrStateDesc
1851     else # We'll have to use the Generic Offset/State description strings
1852         # Use SEL EB1 [3:0] bits as a Generic Offset/State value. Use
1853         # this value, and the Event/ReadingTypeCode , to index into
1854         # Table 36-1 to find a Generic Offset/State description string.
1855         strOffsetOrStateDesc = strEDir + <EStr(<EB1[3:0] into Table36-1>)>
1856         return strOffsetOrStateDesc
1857     else # This isn't a SEL Reading Type code that the IPMI-to-CIM
1858         # mapping supports. Skip this SEL record. It will not
1859         # be mapped to an instance of the LogRecord class. In other
1860         # words, the IPMI-to-CIM mapping ignores OEM type SEL
```


1861 # Records.

1862 **4.5.2.9 CIM_LogRecord.Description**

1863 (string)(<CIM_Sensor.Name>) || “: ” || (string)(EStr(CIM_Sensor.SensorType)) || “ for ” ||
1864 EStr(<IPMI_SDR_EntityID>) || “ “ || (string)(<IPMI_SDR_EntityInstanceNumber>) || “; ”
1865 EStr(<SEL Generic or Sensor-Specific State Code>)

1866 See the procedure under CIM_LogRecord.Caption for a procedural description of the following
1867 term:

1868 EStr(<SEL Generic or Sensor-Specific State Code>).

1869 A SEL record’s associated CIM_Sensor instance (whether that is a NumericSensor or a Sensor
1870 instance) can be found using the SEL Record’s GeneratorID (bytes 8-9) and Sensor Number
1871 (byte 12). See Table 26-1 for details. This SEL data maps to the CIM_Sensor.DeviceID key
1872 value (as defined in the IPMI-to-CIM sensor mapping document). Once this Sensor key string is
1873 extracted from the SEL entry in question, the corresponding CIM_Sensor instance can be found.

1874 Behind the scenes, the corresponding CIM_Sensor maps to a corresponding IPMI SDR record
1875 (or either type 1 or type 2). This indirectly corresponding SDR record must also be found in
1876 order to extract the IPMI_SDR_EntityID and IPMI_SDR_EntityInstanceNumber mentioned
1877 above.

1878 We refer to the CIM_Sensor instance above because CIM_Sensor is a common base class for
1879 both NumericSensor and Sensor (as “trivially derived” for this IPMI-to-CIM mapping).

1880 **5 IPMI Commands Mapping**

1881 **5.1 Introduction**

1882 The IPMI Commands Mapping allows the execution of a controlled subset of available IPMI
1883 commands using a CIM_CommandService class derived from CIM_Service.

1884 **5.2 Mapping Requirements**

1885 **5.2.1 IPMI Commands Limitations**

1886 The IPMI Commands Mapping is the CIM object model to the IPMI interface mapping needed
1887 to issue and receive OEM “raw” messages to and from an embedded Baseboard Management
1888 Controller or Service Processor with IPMI compliant firmware. The class modeled in this
1889 profile is related to the IPMI BMC Messages as detailed in the IPMI Specification.

1890 This model is only intended to map the execution of IPMI OEM commands. Execution of
1891 standard IPMI commands is NOT supported by this class. Standard commands are handled
1892 internally by the Sensor and Log Mapping implementations described in this document.

1893 Also note that one instance of the CIM_CommandService class MUST be created for each IPMI
1894 capable microcontroller, in order to ensure that IPMI OEM commands destined for satellite
1895 controllers are routed correctly

1896 **5.3 CIM_CommandService Mapping**

1897 **5.3.1 CIM_CommandService.Methods**

1898 **5.3.1.1 Method: ExecuteOEMCommand**

```
1899     uint32 ExecuteOEMCommand(  
1900         [IN] uint8 NetFN,  
1901         [IN] uint8 CommandIdentifier,  
1902         [IN] uint8 RequestPayload[],  
1903         [IN ( false ), OUT] uint8 ResponseNetFN,  
1904         [IN ( false ), OUT] uint8 ResponseCommandIdentifier,  
1905         [IN ( false ), OUT] uint8 CompletionCode,  
1906         [IN ( false ), OUT] uint8 ResponsePayload[])
```

1907 MUST check for valid range of command request, as only OEM IPMI commands are
1908 supported by this method. The request is issued to the BMC only if successfully verified.
1909 Valid input parameter ranges for OEM commands are as follows:

- 1910 1. NetFN in range 30h through 3Fh, as described in IPMI v1.5 specification, section 5.1,
1911 table 5.1
- 1912 2. CommandIdentifier in range 00h through FDh., as described in IPMI v1.5
1913 specification, section 5.0

1914 3. When RequestPayload is assembled with NetFN, CommandIdentifier, and other
 1915 request message fields, the total message length must conform to limits specified in
 1916 IPMI V1.5 specification, Sections 6.13 and Appendix D.

1917 The appropriate message bytes in the resulting response are assigned to the ResponseNetFN,
 1918 ResponseCommandIdentifier, CompletionCode, and ResponsePayload[] output parameters .
 1919 Note that message CompletionCode output parameter MUST support all valid values
 1920 described in IPMI v1.5 specification, section 5.2, table 5.2, including device-specific OEM
 1921 completion codes ranges 01h – 7eh.

1922 Return value:

- 1923 0 – Completed with no error
- 1924 1 – Not Supported
- 1925 2 – Unspecified Error
- 1926 3 – Timeout
- 1927 4 – Failed
- 1928 5 – Invalid Parameter
- 1929 6 – NetFN Out Of Range
- 1930 7 – CommandIdentifier Out Of Range
- 1931 8 – RequestPayload too long

1932

1933 5.3.2 CIM_CommandService.Properties

1934 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
 1935 properties that MAY be present. For example, there may be properties that are only filled out in
 1936 special configurations of the CIM Schema.

1937 Properties that listed in the Required section in Table 5 below MUST be implemented as defined
 1938 in the subsection following this table unless the Notes column for each property specifies the
 1939 defining reference.

1940 **Table 5 CIM_CommandService Class Properties**

Required Properties	Notes
CreationClassName	Key:
Name	Key:
SystemCreationClassName	Key:
SystemName	Key:
IANA	Key:
<i>Recommended Properties</i>	Notes
Caption	
Description	
Started	Not Defined.
StartMode	Not Defined.
Status	Not Defined.
InstallDate	Not defined.

1941

<i>Recommended Properties</i>	<i>Notes</i>
I	

1942

1943 The following subsections describe procedures for generating values for various properties of
1944 CIM classes required in this profile.

1945 **5.3.2.1 CIM_CommandService.SystemCreationClassName**

1946 Formulate as a string:

1947 <ComputerSystem.Name of ComputerSystem instance of the one associated BMC>

1948 This is the scoping system's CreationClassName.

1949 **5.3.2.2 CIM_CommandService.SystemName**

1950 Name of the system that is hosting the CommandService

1951 **5.3.2.3 CIM_CommandService.CreationClassName**

1952 Fixed value of "CommandService"

1953 **5.3.2.4 CIM_CommandService.Name**

1954 Name of IPMI controller to which commands will be routed, such as "Primary BMC", "OEM
1955 satellite controller 1", "OEM satellite controller 2", etc.

1956 **5.3.2.5 CIM_CommandService.IANA**

1957 IANA ID for microcontroller

1958 **5.3.2.6 CIM_CommandService.Caption**

1959 Fixed value of "IPMI Command Service"

1960 **5.3.2.7 CIM_CommandService.Description**

1961 Fixed value of "IPMI Command Service"

1962

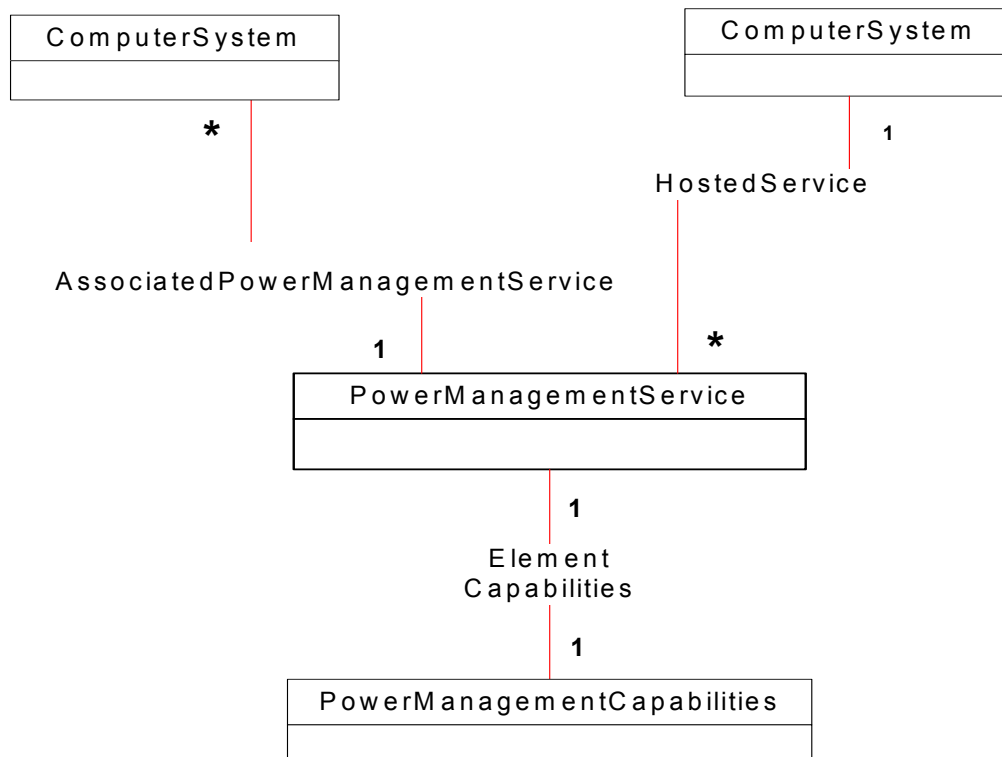
1963 6 IPMI Power Control Mapping

1964 6.1 Introduction

1965 The IPMI Power Control Mapping extends the management capability of the IPMI to CIM
 1966 mapping by adding the capability to manage the power state of the managed system and the
 1967 BMC via IPMI represented as a CIM_PowerManagementService class.

1968 The following class diagram describes the CIM classes involved in representing IPMI power
 1969 control by the CIM_PowerManagementService class. The ComputerSystem classes are shown
 1970 in this diagram to illustrate the way the CIM_PowerManagementService class is associated with
 1971 overall class ecosystems. This is in preparation for interoperability with forthcoming profiles
 1972 from the DMTF and other standards bodies.

Power Control Class Diagram



1973

1974 **Figure 5 Class Diagram – IPMI Computer System Power Management**

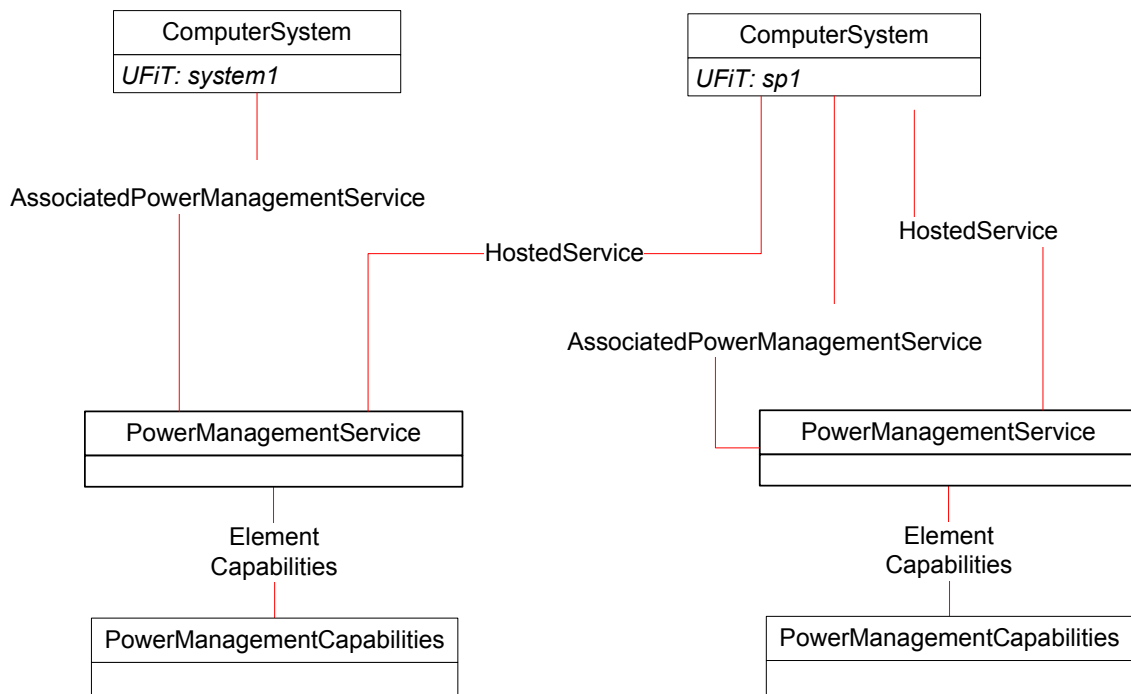
1975 6.2 Instance Diagrams

1976 The IPMI Computer System Power Management Mapping is the CIM object model to IPMI
 1977 interface mapping needed to manage the managed or host system and the BMC power state.

1978 The following instance diagram represents that instantiation of 2
 1979 CIM_PowerManagementService classes each responsible for representing the power control
 1980 interface for either the BMC or the host system that the BMC is embedded in. The BMC and the
 1981 host system are represented by instance of the CIM_ComputerSystem class.

**Power Control
 Instance Diagram:**

**Monolithic Server
 with Service Processor**



1982

1983 **Figure 6 Instance Diagram – IPMI Computer System Power Management**

1984 **6.3 Mapping Requirements**

1985 **6.4 CIM_PowerManagementService Mapping**

1986 **6.4.1 CIM_PowerManagementService Methods**

1987 **6.4.1.1 Method: PowerManagementService.SetPowerState()**

```
1988     uint32 SetPowerState( uint16 PowerState,
1989                          CIM_ManagedElement REF ManagedElement,
1990                          datetime Time
1991                          );
```

1992 The CIM SetPowerState() method PowerState parameter is an enumerated value that
 1993 corresponds to the IPMI Chassis Control command input parameter according to the following
 1994 table:

CIM SetPowerState().PowerState Parameter	IPMI Chassis Command Parameter
Off-Hard	0h
On	01h
PowerCycle (Off-Hard)	02h
PowerCycle (Off-Soft)	03h
VendorDefined (for NMI)	04h
Off-Soft	05h

1995

1996 **6.4.1.2 Method: PowerManagementService.StartService()**

1997 This method MUST return with a value of 1 or “Not Supported”.

1998 **6.4.1.3 Method: PowerManagementService.StopService()**

1999 This method MUST return with a value of 1 or “Not Supported”.

2000 **6.4.1.4 Method: PowerManagementService.RequestStateChange()**

2001 `uint32 RequestStateChange (`
 2002 `[IN] uint16 RequestedState,`
 2003 `[IN] datetime TimeoutPeriod)`

2004 Return value:

2005 0 – Completed with no error
 2006 1 – Not Supported
 2007 2 – Unspecified Error
 2008 3 – Timeout
 2009 4 – Failed
 2010 5 – Invalid Parameter
 2011 6..0xFFFF – DMTF_Reserved
 2012 0x1000..0x7FFF – Method_Reserved
 2013 0x8000.. – Vendor_Reserved

2014 **6.4.2 CIM_PowerManagementService.Properties**

2015 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
 2016 properties that MAY be present. For example, there may be properties that are only filled out in
 2017 special configurations of the CIM Schema.

2018 Properties that listed in the Required section in Table 6 below MUST be implemented as defined
 2019 in the subsections following this table unless the Notes column for each property specifies the
 2020 defining reference.

2021

Table 6 CIM_PowerManagementService Class Properties

Required Properties	Notes
CreationClassName	Key
Name	Key
<i>Recommended Properties</i>	Notes
InstallDate	
StatusDescriptions	
Caption	
Description	
ElementName	

2022

2023

2024 The following subsections describe procedures for generating values for various properties of
 2025 CIM classes required in this profile.

2026 **6.4.2.1 CIM_PowerManagementService.CreationClassName**

2027 Per the DMTF definition of the class property: The name of the class or the subclass used in the
 2028 creation of an instance of the log for with this record is a part. This will typically be
 2029 “CIM_PowerManagemetnService”.

2030 Formulate as a string:

2031 <“CIM_PowerManagementService” or the name of the subclass created>

2032 **6.4.2.2 CIM_PowerManagementService.Name**

2033 Fixed value of “IPMI Power Service”

2034 **6.5 CIM_PowerManagementCapabilities Mapping**

2035 **6.5.1 CIM_PowerManagementCapabilities Methods**

2036 None Defined.

2037 **6.5.2 CIM_PowerManagementCapabilities Properties**

2038 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
 2039 properties that MAY be present according to t he guiding DMTF profiles.

2040 Properties that listed in the Required section in Table 7 below MUST be implemented as defined
 2041 in the subsection following this table unless the Notes column for each property specifies the
 2042 defining reference.

2043

Table 7 CIM_PowerManagementCapabilities Class Properties

2044

Required Properties	Notes
PowerCapabilities	
Recommended Properties	Notes
Caption	
Description	
ElementName	
OtherPowerCapabilitiesDescriptions	

2045 6.5.2.1 CIM_PowerManagementCapabilities.PowerCapabilities

2046 Fixed value of 3 (for the enumeration "Power State Settable").

2047 6.6 CIM_AssociatedPowerManagementService Mapping**2048 6.6.1 CIM_AssociatedPowerManagementService Methods**

2049 None Defined.

2050 6.6.2 CIM_AssociatedPowerManagementService Properties

2051 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
 2052 properties that MAY be present. For example, there may be properties that are only filled out in
 2053 special configurations of the CIM Schema.

2054 Properties that are listed in the Required section in Table 8 below MUST be implemented as
 2055 defined in the subsections following this table unless the Notes column for each property
 2056 specifies the defining reference.

2057 Table 8 CIM_AssociatedPowerManagementService Class Properties

Required Properties	Notes
ServiceProvided	REF CIM_PowerManagementService {key, *}
UserOfService	REF ManagedElement {key, *}
PowerState	Values { "Other", "On", "Sleep - Light", "Sleep - Deep", "Off - Hard", "Hibernate (Off - Soft)", "Off - Soft", "DMTF Reserved", "Vendor Specific" },
OtherPowerState	
PowerOnTime	

2058

<i>Recommended Properties</i>	Notes
Caption	
Description	

2059 **6.6.2.1 CIM_AssociatedPowerManagementService.PowerState**2060 **6.6.2.2 CIM_AssociatedPowerManagementService.OtherPowerState**2061 **6.6.2.3 CIM_AssociatedPowerManagementService.PowerOnTime**

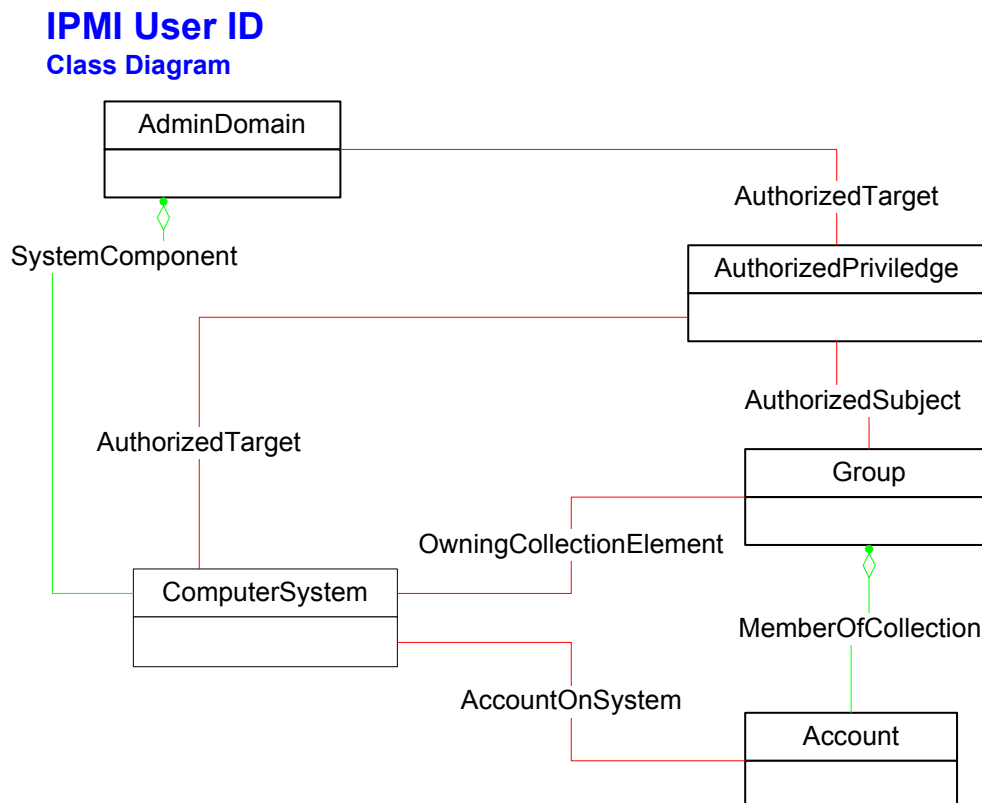
2062

2063 7 IPMI User ID Management Mapping

2064 7.1 Introduction

2065 The IPMI User ID Management Mapping extends the management capability of the IPMI to
 2066 CIM mapping by adding the capability to manage the User ID, passwords, and privileges of the
 2067 BMC.

2068 The following class diagram describes the CIM classes involved in representing IPMI User ID's,
 2069 passwords, and privileges. The AdminDomain and ComputerSystem classes are shown in this
 2070 diagram to illustrate the way the various classes involved in user id management are associated
 2071 with overall class ecosystems. This is in preparation for interoperability with forthcoming
 2072 profiles from the DMTF and other standards bodies.



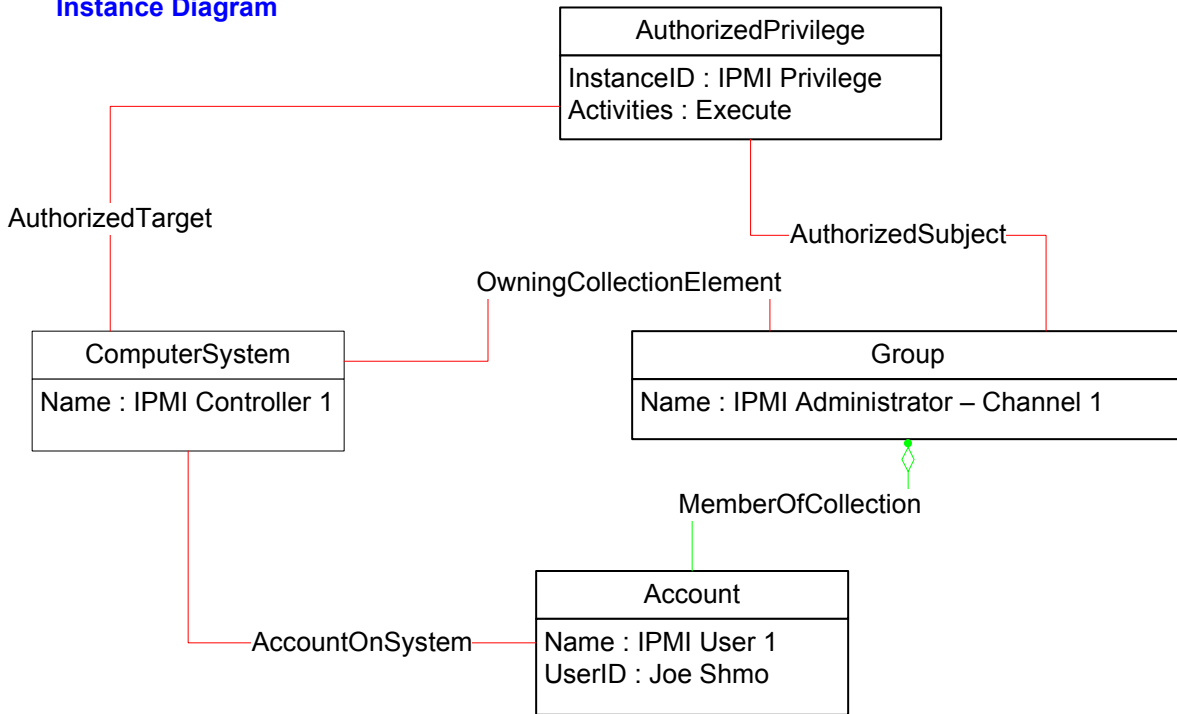
2073

2074 **Figure 7 Class Diagram – IPMI User ID Management**

2075 7.2 Instance Diagrams

2076 The IPMI User ID Management Mapping is the CIM object model to IPMI interface mapping
 2077 needed to manage the BMC User ID's. The BMC and the host system are represented by
 2078 instance of the CIM_ComputerSystem class.

**IPMI User ID
Instance Diagram**



2079

2080

Figure 8 Instance Diagram – IPMI User ID Management

2081 7.3 Mapping Requirements

2082 7.3.1 Mapping IPMI User ID's

2083 The IPMI User ID's are to be treated as slots that always exist irregardless of whether the
 2084 username has been defined or whether the User ID is active. The implementation will
 2085 instantiate the CIM_Account instance for the number of User ID's the IPMI implementation
 2086 supports.

2087 7.3.2 Mapping IPMI Access Privileges

2088 Instances of CIM_Group are created for each of the possible privilege levels (eg Administrator,
 2089 Operator, User, Callback, OEM, No Access) per Channel that an IPMI User ID may be granted.
 2090 A specific CIM_Account instance may be a member of several groups, in which case the highest
 2091 level access/execute privilege is in force. The access/execute privilege level in descending order
 2092 is:

Group Privilege Level Order – Channel 1
Administrator – Channel 1
No-Access – Channel 1
Operator – Channel 1

Group Privilege Level Order – Channel 1
User – Channel 1
Call-Back – Channel 1
OEM – Channel 1

2093

2094 For every channel available in the implementation, a set of instances of CIM_Group
 2095 corresponding to the possible privilege levels for that channel will need to be instantiated.

2096 Example for an implementation with 2 channels: The following 12 instances of CIM_Group are
 2097 required:

Groups for Channel 1	Groups for Channel2
Administrator – Channel 1	Administrator – Channel 2
No-Access – Channel 1	No-Access – Channel 2
Operator – Channel 1	Operator – Channel 2
User – Channel 1	User – Channel 2
Call-Back – Channel 1	Call-Back – Channel 2
OEM – Channel 1	OEM – Channel 2

2098 7.3.3 Modeling CIM_AuthorizedPrivilege

2099 The CIM_AuthorizedPrivilege is modeled as granting “execute” authorization to the various
 2100 group it is associated with. The implementation MUST instantiate one instance of
 2101 CIM_AuthorizedPrivilege and associate each IPMI group with it.

2102 7.4 CIM_Account Mapping

2103 7.4.1 CIM_Account Methods

2104 None Defined.

2105 7.4.2 CIM_Account Properties

2106 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
 2107 properties that MAY be present. For example, there may be properties that are only filled out in
 2108 special configurations of the CIM Schema.

2109 Properties that listed in the Required section in Table 9 below MUST be implemented as defined
 2110 in the subsections following this table unless the Notes column for each property specifies the
 2111 defining reference.

2112

Table 9 CIM_Account Class Properties

Required Properties	Notes
SystemCreationClassName	Key
SystemName	Key
CreationClassName	Key
Name	Key
UserID	
UserPassword	
OperationalStatus	
StatusDescriptions	

2113

Recommended Properties	Notes
ObjectClass	
Descriptions	
Host	
LocalityName	
OrganizationalName	
OU	
SeeAlso	
UserCertificate	
InstallDate	
HealthState	
Caption	
Description	
ElementName	

2114

2115 The following subsections describe procedures for generating values for various properties of
 2116 CIM classes required in this profile.

2117 **7.4.2.1 CIM_Account.SystemCreationClassName**

2118 Per the DMTF definition of the class property: The scoping System's CreationClassName. In all
 2119 cases this will be the class "CIM_ComputerSystem".

2120 Formulate as a string:

2121 <"CIM_ComputerSystem">

2122 **7.4.2.2 CIM_Account.SystemName**

2123 Formulate as a string:

2124 <ComputerSystem.Name of ComputerSystem instance of the one associated BMC>

2125 The System instance that represents the associated management controller has a Name property.
2126 This be found and then copied in here. Since there is but one instance of a management
2127 controller for most IPMI-compliant devices/systems, this is often quite easy to do.

2128 **7.4.2.3 CIM_Account.CreationClassName**

2129 Per the DMTF definition of the class property: The name of the class or the subclass used in the
2130 creation of an instance of the log for with this record is a part. This will typically be
2131 “CIM_Account”.

2132 Formulate as a string:

2133 <”CIM_Account” or the name of the subclass created>

2134 **7.4.2.4 CIM_Account.Name**

2135 IPMI User ID slot number.

2136 Formulate as a string:

2137 ”IPMI User” || <IPMI User ID>

2138 Where the IPMI User ID is a number from 1 to N where N is the number of User ID’s supported
2139 by the IPMI implementation. An instance of CIM_Account MUST be instantiated for every
2140 possible used, including those cases where the UserName is Null.

2141 **7.4.2.5 CIM_Account.UserID**

2142 IPMI User Name.

2143 Formulate as a string:

2144 <IPMI User Name>

2145 The IPMI User Name is retrieved from the IPMI command GetUsername().

2146 **7.4.2.6 CIM_Account.UserPassword**

2147 IPMI Password.

2148 This is always Null for read operations.

2149 **7.4.2.7 CIM_Account.OperationalStatus**

2150

2151 **7.4.2.8 CIM_Account.StatusDescriptions**

2152 **7.5 CIM_Group Mapping**

2153 **7.5.1 CIM_Group Methods**

2154 None Defined.

2155 **7.5.2 CIM_Group Properties**

2156 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
 2157 properties that MAY be present according to the guiding DMTF profiles.

2158 Properties that listed in the Required section in Table 10 below MUST be implemented as
 2159 defined in the subsection following this table unless the Notes column for each property specifies
 2160 the defining reference.

2161 **Table 10 CIM_Group Class Properties**

Required Properties	Notes
CreationClassName	Key
Name	Key

2162

Recommended Properties	Notes
BusinessCategory	
CommonName	
Caption	
Description	
ElementName	

2163 **7.5.2.1 CIM_Group.CreationClassName**

2164 Per the DMTF definition of the class property: The name of the class or the subclass used in the
 2165 creation of an instance of the log for with this record is a part. This will typically be
 2166 “CIM_Group”.

2167 Formulate as a string:

2168 <”CIM_Group” or the name of the subclass created>

2169 **7.5.2.2 CIM_Group.Name**

2170 Instances of CIM_Group MUST be created for each of the following IPMI user level privileges.

2171 Formulate as a string the following IPMI user level privileges:

2172 IPMI Administrator

2173	IPMI Operator
2174	IPMI User
2175	IPMI Callback
2176	IPMI OEM
2177	IPMI No Access

2178 7.6 CIM_AuthorizedPrivilege Mapping

2179 7.6.1 CIM_AuthorizedPrivilege Methods

2180 None Defined.

2181 7.6.2 CIM_AuthorizedPrivilege Properties

2182 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
 2183 properties that MAY be present. For example, there may be properties that are only filled out in
 2184 special configurations of the CIM Schema.

2185 Properties that listed in the Required section in Table 11 below MUST be implemented as
 2186 defined in the subsections following this table unless the Notes column for each property
 2187 specifies the defining reference.

2188 **Table 11 CIM_AuthorizedPrivilege Class Properties**

Required Properties	Notes
InstanceID	Key
PrivilegeGranted	Boolean
Activities	
ElementName	
Recommended Properties	Notes
ActivityQualifiers	
QualifierFormats	
Caption	
Description	

2190 7.6.2.1 CIM_AuthorizedPrivilege.InstanceID

2191 Formulate as a string:

2192 “IPMI Privilege”

2193 **7.6.2.2 CIM_AuthorizedPrivilege.PrivilegeGranted**

2194 Formulate as a Boolean:

2195 Yes

2196 **7.6.2.3 CIM_AuthorizedPrivilege.Activities**

2197 Formulate as an integer enumeration:

2198 7 (Enumeration = "Execute")

2199 **7.6.2.4 CIM_AuthorizedPrivilege.ElementName**

2200 Formulate as a string:

2201 "IPMI Execute Privilege"

2202 8 IPMI BMC Mapping

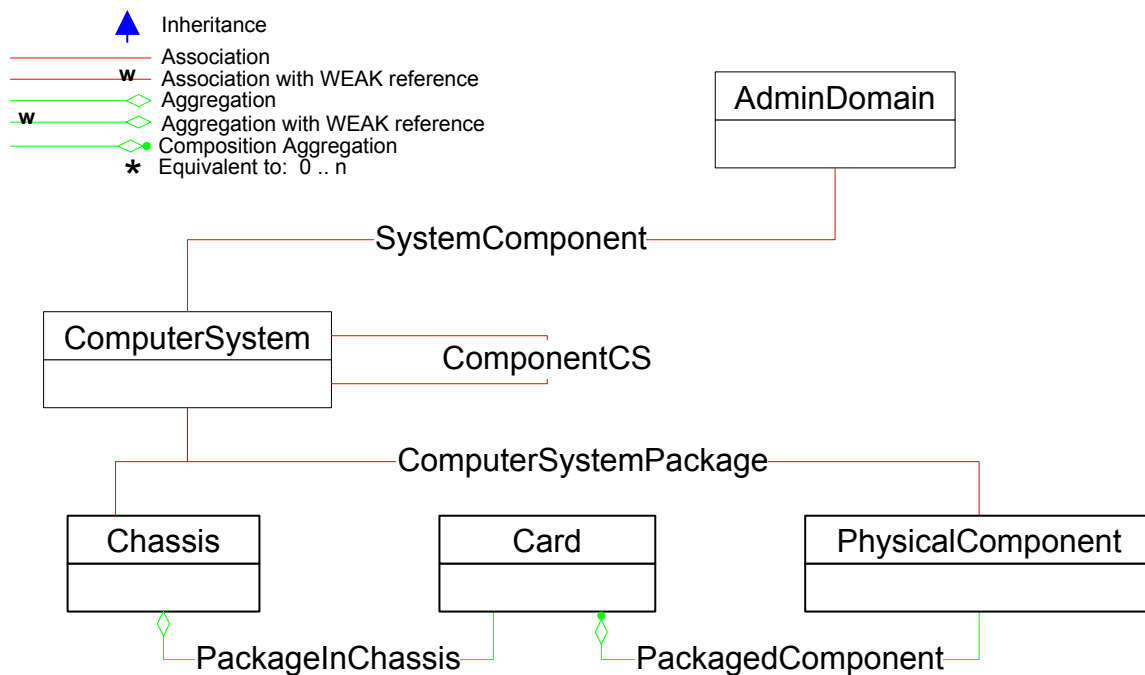
2203 8.1 Introduction

2204 The IPMI BMC Mapping extends the management capability of the IPMI to CIM mapping by
 2205 adding the capability to model and manage the BMC as a Manageability Access Point (MAP).
 2206 Also modeled is the managed system that the BMC is embedded in.

2207 The following class diagram describes the CIM classes involved in representing BMC admin
 2208 domain, the BMC itself and the managed system the BMC is embedded in. This is in preparation
 2209 for interoperability with forthcoming profiles from the DMTF and other standards bodies.

2210 The AdminDomain class is for supporting DMTF SMASH CLP addressability and is consistent
 2211 with the modeling approach.

IPMI BMC MAP



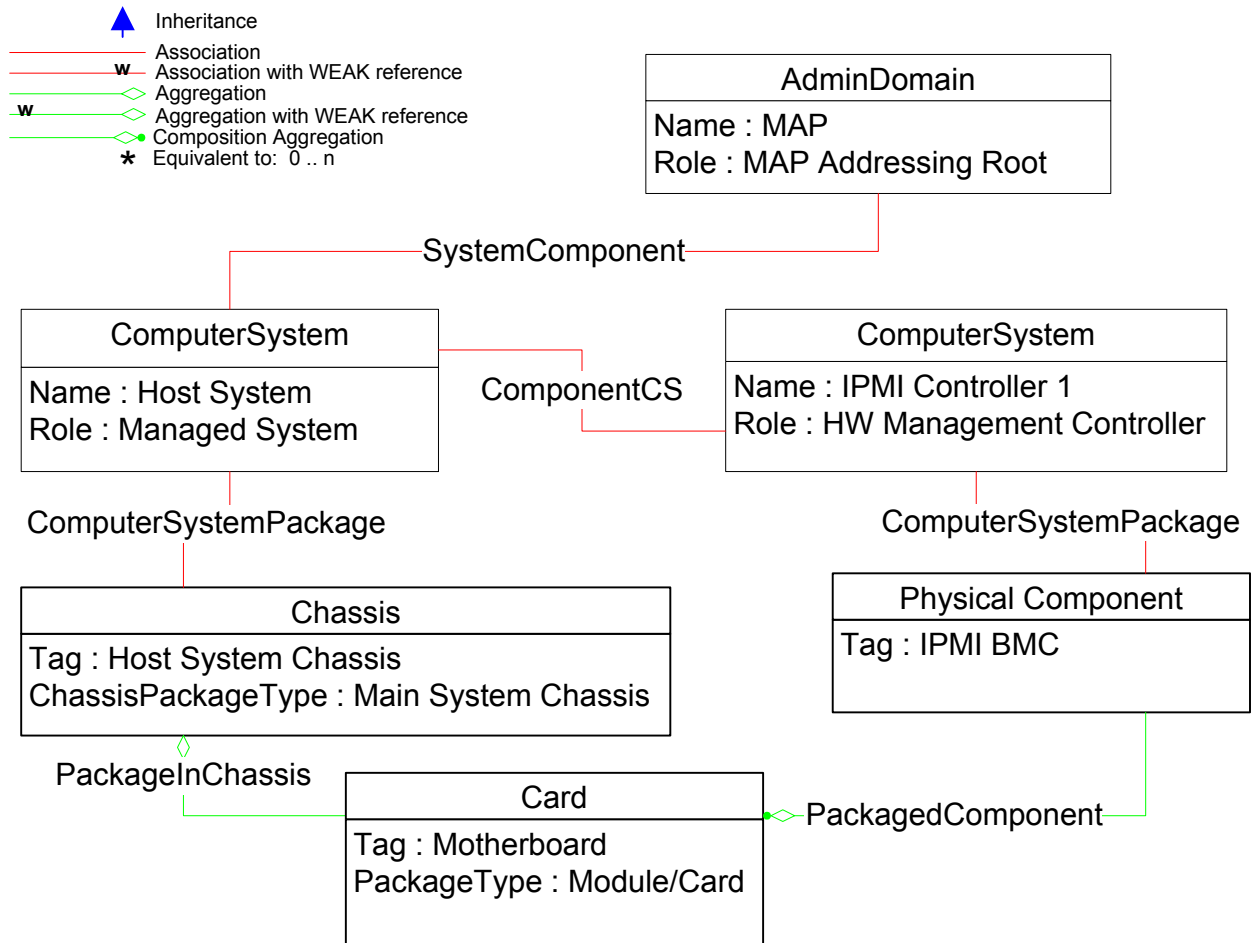
2212

2213 **Figure 9 Class Diagram – IPMI BMC MAP Management**

2214 8.2 Instance Diagrams

2215 The IPMI BMC MAP Management mapping is the CIM object model to IPMI interface mapping
 2216 needed to represent and manage BMC.

IPMI BMC MAP Instance



2217

2218

Figure 10 Instance Diagram – IPMI BMC MAP Management

2219

8.3 Mapping Requirements

2220

The implementation **MUST** instantiate a single instance of CIM_AdminDomain in the implementation namespace and **MUST** instantiate two (2) instances of CIM_ComputerSystem.

2221

The first instance of CIM_ComputerSystem is to represent the BMC and the second instance is to represent the host or managed system.

2222

2223

2224

2225

Physical classes such as CIM_Chassis and CIM_Card **SHOULD** be instantiated if the implementation has FRU and/or other host system inventory data, including the host system name.

2226

2227

2228 8.4 CIM_AdminDomain Mapping

2229 8.4.1 CIM_AdminDomain Methods

2230 8.4.1.1 Method: AdminDomain.RequestStateChange()

```
2231 uint32 RequestStateChange (
2232     [IN] uint16 RequestedState,
2233     [IN] datetime TimeoutPeriod)
```

2234 Return value:

2235 0 – Completed with no error

2236 1 – Not Supported

2237 2 – Unspecified Error

2238 3 – Timeout

2239 4 – Failed

2240 5 – Invalid Parameter

2241 6..0xFFFF – DMTF_Reserved

2242 0x1000..0x7FFF – Method_Reserved

2243 0x8000.. – Vendor_Reserved

2244 8.4.2 CIM_AdminDomain.Properties

2245 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
2246 properties that MAY be present. For example, there may be properties that are only filled out in
2247 special configurations of the CIM Schema.

2248 Properties that are listed in the Required section in Table 6 below MUST be implemented as
2249 defined in the subsections following this table unless the Notes column for each property
2250 specifies the defining reference.

2251 **Table 12 CIM_AdminDomain Class Properties**

Required Properties	Notes
CreationClassName	Key
Name	Key
Recommended Properties	Notes
NameFormat	
PrimaryOwnerName	
PrimaryOwnerContact	
Roles	
EnabledState	
OtherEnabledState	
RequestedState	
EnabledDefault	

<i>Recommended Properties</i>	<i>Notes</i>
TimeOfLastStateChange	
InstallDate	
OperationalStatus	
StatusDescriptions	
HealthState	
Caption	
Description	
ElementName	

2253

2254 The following subsections describe procedures for generating values for various properties of
2255 CIM classes required in this profile.

2256 **8.4.2.1 CIM_AdminDomain.CreationClassName**

2257 Per the DMTF definition of the class property: The name of the class or the subclass used in the
2258 creation of an instance of the log for with this record is a part. This will typically be
2259 “CIM_AdminDomain”.

2260 Formulate as a string:

2261 <“CIM_AdminDomain” or the name of the subclass created>

2262 **8.4.2.2 CIM_AdminDomain.Name**

2263 Fixed value of “MAP”

2264 **8.5 CIM_ComputerSystem Mapping**

2265 **8.5.1 CIM_ComputerSystem Methods**

2266 **8.5.1.1 Method: ComputerSystem.RequestStateChange()**

```
2267 uint32 RequestStateChange (
2268     [IN] uint16 RequestedState,
2269     [IN] datetime TimeoutPeriod)
```

2270 Return value:

2271 0 – Completed with no error

2272 1 – Not Supported

2273 2 – Unspecified Error

2274 3 – Timeout

2275 4 – Failed

2276 5 – Invalid Parameter

2277 6..0x0FFF – DMTF_Reserved
 2278 0x1000..0x7FFF – Method_Reserved
 2279 0x8000.. – Vendor_Reserved

2280 8.5.2 CIM_ComputerSystem Properties

2281 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
 2282 properties that MAY be present according to the guiding DMTF profiles.

2283 Properties that listed in the Required section in Table 7 below MUST be implemented as defined
 2284 in the subsection following this table unless the Notes column for each property specifies the
 2285 defining reference.

2286 **Table 13 CIM_ComputerSystem Class Properties**

Required Properties	Notes
Name	KEY
CreationClassName	KEY
Dedicated	
Roles	
EnabledState	
HealthState	
OperationalStatus	
<i>Recommended Properties</i>	Notes
NameFormat	
OtherIdentifyingInfo	
IdentifyingDescriptions	
OtherDedicatedDescriptions	
ResetCapability	
PrimaryOwnerName	
PrimaryOwnerContact	
RequestedState	
OtherEnabledState	
EnabledDefault	
TimeOfLastStateChange	
InstallDate	
StatusDescriptions	
Caption	
Description	
ElementName	

2288

2289 The following sections may describe the values for two instances of CIM_ComputerSystem.
2290 The values for the instance of CIM_ComputerSystem that represent the BMC are labeled with a
2291 “**BMC:**” and the value definitions for the instance that represent the managed system are labeled
2292 “**Host:**”.

2293 **8.5.2.1 CIM_ComputerSystem.Name**

2294 **BMC:**

2295 Concatenation of “IPMI Controller” and BMC device ID (Get Device ID response byte 2)

2296 Formulate as a string:

2297 <“IPMI Controller” || string (Get Device ID response byte 2)>

2298 Example Value: “IPMI Controller 32”

2299 **Host:**

2300 The goal for the value of the Name property for the instance of CIM_ComputerSystem
2301 representing the IPMI host or managed system is to provide a correlatable identifier that can be
2302 used by applications to correlate representations from different manageability access points. At
2303 present, due to various versions of IPMI being available in the field and a general lack of
2304 consensus in the industry as to what correlatable identifier should be predominant, the value
2305 formulation presented here is an attempt to define a formulation that can be generated across
2306 various situations.

2307 The general format of the Name property shall be:

2308 < (NameValue) || “:” ||

2309 string (GUID as retrieved from Get System GUID IPMI Command)>

2310 Where NameValue is generated in the following fashion:

2311 IF the implementation resides in-band in the OS resident on the system containing the BMC, the
2312 implementation should use the value of the Name property in the instance of
2313 CIM_ComputerSystem generated by OS instrumentation and representing the managed system
2314 for NameValue.

2315 Else IF the MultiRecord Area is available in the system FRU and the Management Access
2316 Record (Type 0x03) Sub-Record Type (0x02) System Name string is populated, use this value
2317 for NameValue.

2318 Else IF the IPMI interface supports the IPMI 2.0 command Get System Info, use the value
2319 returned in the System Name parameter for NameValue.

2320 Else use the following value for NameValue:

2321 < “Unknown” || “.” || “IPMI BMC Device ID” || “.” ||

2322 string (Device ID as retrieved from the Get Device ID IPMI command)>

2323 **8.5.2.2 CIM_ComputerSystem.CreationClassName**

2324 Per the DMTF definition of the class property: The name of the class or the subclass used in the
2325 creation of an instance of the log for with this record is a part. This will typically be
2326 “CIM_ComputerSystem”.

2327 Formulate as a string:

2328 <”CIM_ComputerSystem” or the name of the subclass created>

2329 **8.5.2.3 CIM_ComputerSystem.Dedicated**

2330 **BMC:**

2331 For the instance of CIM_ComputerSystem representing the IPMI BMC:

2332 This is a fixed value: 14 (enumerated as “Management Controller”)

2333 **Host:**

2334 The value of the Dedicated property for the managed system is left up to the implementation to
2335 select the appropriate value from the enumeration value/value maps defined in the
2336 CIM_ComputerSystem MOF. However, the when implementation is representing a general
2337 purpose server it should use the value of:

2338 0 (enumerated as “Not Dedicated”)

2339 **8.5.2.4 CIM_ComputerSystem.Roles**

2340 This property is writeable and may be changed by the user/application.

2341 **BMC:**

2342 For the instance of CIM_ComputerSystem representing the IPMI BMC:

2343 The initial value: “Hardware Management Controller”

2344 **Host:**

2345 For the instance of CIM_ComputerSystem representing the managed system:

2346 The initial value: “Managed System”

2347 **8.5.2.5 CIM_ComputerSystem.EnabledState**

2348 **8.5.2.6 CIM_ComputerSystem.HealthState**

2349 **8.5.2.7 CIM_ComputerSystem.OperationalStatus**

2350 **8.5.2.8 CIM_ComputerSystem.NameFormat**

2351 Format of the Name string generated by this mapping:

2352 This is a fixed value of “Other”

2353 **8.5.2.9 CIM_ComputerSystem.ElementName**

2354 **BMC:**

2355 Concatenation of “IPMI Controller” and BMC device ID (Get Device ID response byte 2)

2356 Formulate as a string:

2357 <>“IPMI Controller” || string (Get Device ID response byte 2)>

2358 Example Value: “IPMI Controller 32”

2359 **Host:**

2360 The goal for the value of the Name property for the instance of CIM_ComputerSystem
 2361 representing the IPMI host or managed system is to provide a correlatable identifier that can be
 2362 used by applications to correlate representations from different manageability access points. At
 2363 present, due to various versions of IPMI being available in the field and a general lack of
 2364 consensus in the industry as to what correlatable identifier should be predominant, the value
 2365 formulation presented here is an attempt to define a formulation that can be generated across
 2366 various situations.

2367 The general format of the Name property shall be:

2368 < (NameValue) || “:” ||

2369 string (GUID as retrieved from Get System GUID IPMI Command)>

2370 Where NameValue is generated in the following fashion:

2371 IF the implementation resides in-band in the OS resident on the system containing the BMC, the
 2372 implementation should use the value of the Name property in the instance of
 2373 CIM_ComputerSystem generated by OS instrumentation and representing the managed system
 2374 for NameValue.

2375 Else IF the MultiRecord Area is available in the system FRU and the Management Access
 2376 Record (Type 0x03) Sub-Record Type (0x02) System Name string is populated, use this value
 2377 for NameValue.

2378 Else IF the IPMI interface supports the IPMI 2.0 command Get System Info, use the value
 2379 returned in the System Name parameter for NameValue.

2380 Else use the following value for NameValue:

2381 < “Unknown” || “.” || “IPMI BMC Device ID” || “.” ||

2382 string (Device ID as retrieved from the Get Device ID IPMI command)>

2383 **8.5.2.10 CIM_ComputerSystem.IdentifyingDescriptions**

2384

2385 Fixed value as: “Manufacturer ID-Product ID”

2386 **8.5.2.11 CIM_ComputerSystem.OtherIdentifyingInfo**

2387 Concatenation of BMC Manufacturer ID (Get Device ID response byte 8-10) and BMC Product
 2388 ID(Get Device ID response byte 11-12)

2389 Formulate as a string:

2390 < string (Get Device ID response byte 8-10) ||

2391 string (Get Device ID response byte 11-12) >

2392 Example:

2393 11102-12345

2394 **8.5.2.12 CIM_ComputerSystem.Caption**

2395 Concatenation of “BMC ” and its device ID (Get Device ID response byte 2)

2396 Formulate as a string:

2397 < “BMC “ || string (Get Device ID response byte 2) >

2398 Example:

2399 “BMC 32”

2400 **8.5.2.13 CIM_ComputerSystem.Description**

2401 Concatenation of BMC device ID (Get Device ID response byte 2), IPMI version (Get Device ID
 2402 response byte 6), BMC firmware revision (Get Device ID response byte 4-5), BMC auxiliary
 2403 firmware revision (Get Device ID response byte 13-16)

2404 Formulate as a string:

2405 < “BMC “ || string (Get Device ID response byte 2) || “ – IPMI v” ||
2406 string (Get Device ID response byte 6) || “ – fw v” ||
2407 string (Get Device ID response byte 4-5) || “. ” ||
2408 string (Get Device ID response byte 13-16) >
2409 Exmample:
2410 “BMC 32 - IPMI v1.5 - fw v0.30.0.0.0.0”

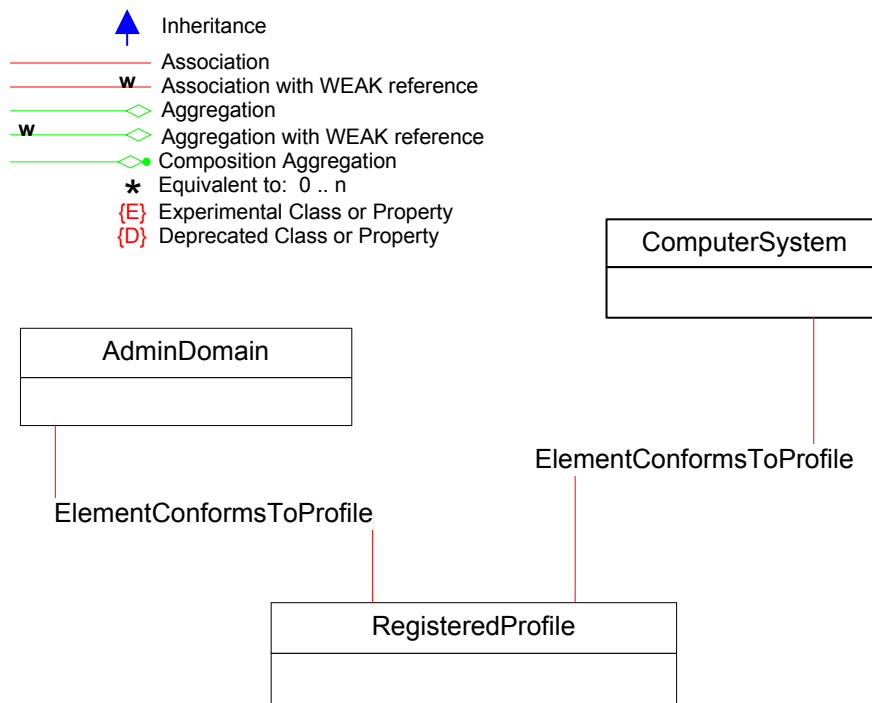
2411 9 IPMI Versioning Mapping

2412 9.1 Introduction

2413 The IPMI Versioning Mapping extends the management capability of the IPMI to CIM mapping
 2414 by adding the capability to model the version of the mapping being implemented

2415 The following class diagram describes the CIM classes involved in representing the IPMI to CIM
 2416 mapping version being implemented. The use of the CIM_RegisteredProfile class to represent
 2417 the version of the IPMI to CIM “Mapping” is not completely aligned with the description of the
 2418 class. However, due to the lack of another class suitable for representing the Mapping version
 2419 information, the mapping will reuse the CIM_RegisteredProfile class. The implementation
 2420 should instantiate the instance of CIM_RegisteredProfile in the implementation namespace so as
 2421 to avoid having to support cross namespace associations.

IPMI Versioning Class Diagram



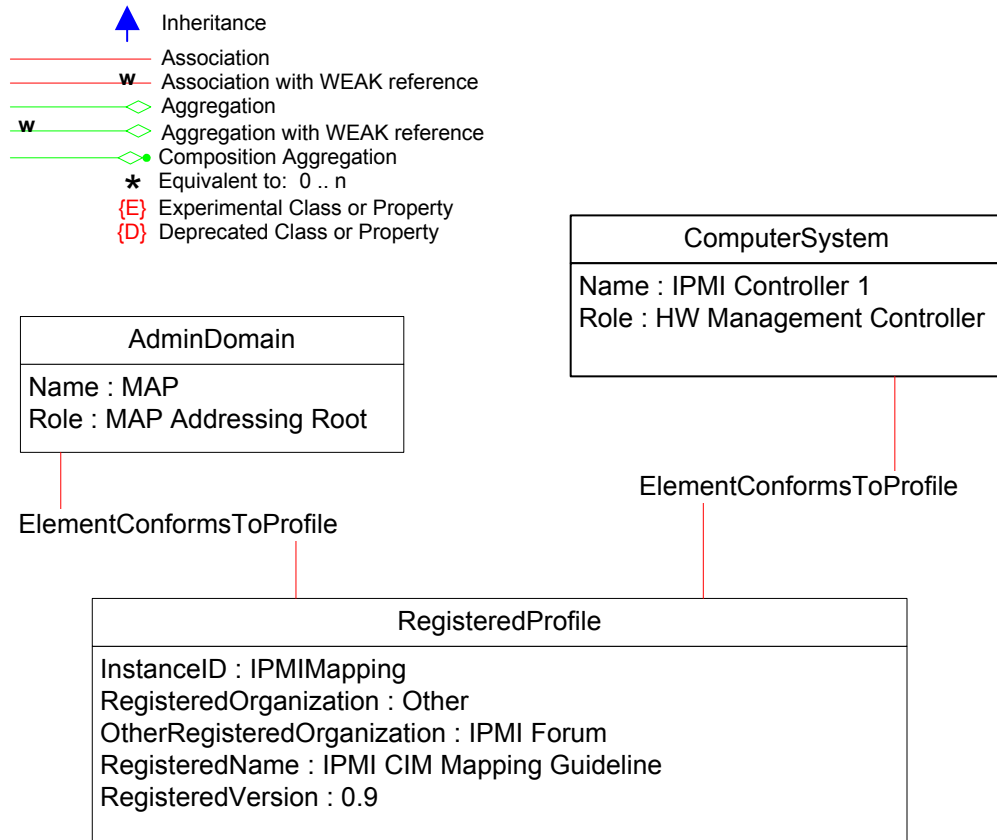
2422

2423 **Figure 11 Class Diagram – IPMI Versioning**

2424 9.2 Instance Diagrams

2425 The IPMI Versioning mapping is the CIM object model representation of the version of the IPMI
 2426 to CIM mapping implemented.

Versioning Instance Diagram



2427

2428

Figure 12 Instance Diagram – IPMI Versioning

2429 9.3 Mapping Requirements

2430 9.4 CIM_RegisteredProfile Mapping

2431 9.4.1 CIM_RegisteredProfile Methods

2432 None.

2433 9.4.2 CIM_RegisteredProfile Properties

2434 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
 2435 properties that MAY be present. For example, there may be properties that are only filled out in
 2436 special configurations of the CIM Schema.

2437 Properties that are listed in the Required section in Table 6 below MUST be implemented as
 2438 defined in the subsections following this table unless the Notes column for each property
 2439 specifies the defining reference.

2440

Table 14 CIM_RegisteredProfile Class Properties

Required Properties	Notes
InstanceID	Key
RegisteredOrganization	
OtherRegisteredOrganization	
RegisteredName	
RegisteredVersion	
<i>Recommended Properties</i>	Notes
AdvertiseTypes	
AdvertiseTypeDescriptions	
Caption	
Description	
ElementName	

2441

2442

2443 The following subsections describe procedures for generating values for various properties of
 2444 CIM classes required in this profile.

2445 **9.4.2.1 CIM_RegisteredProfile.InstanceID**

2446 Formulate as string:

2447 “IPMI : Mapping”

2448 **9.4.2.2 CIM_RegisteredProfile.RegisteredOrganization**

2449 Formulate as uint16:

2450 1 – (Other)

2451 **9.4.2.3 CIM_RegisteredProfile.OtherRegisteredOrganization**

2452 Formulate as string:

2453 “IPMI Forum”

2454 **9.4.2.4 CIM_RegisteredProfile.RegisteredName**

2455 Formulate as string:

2456 “IPMI CIM Mapping Guideline”

2457 **9.4.2.5 CIM_RegisteredProfile.RegisteredVersion**

2458 Formulate as string:

2459

“Version” || <version of this IPMI CIM Mapping Guideline>

2460 10 IPMI Entity Mapping

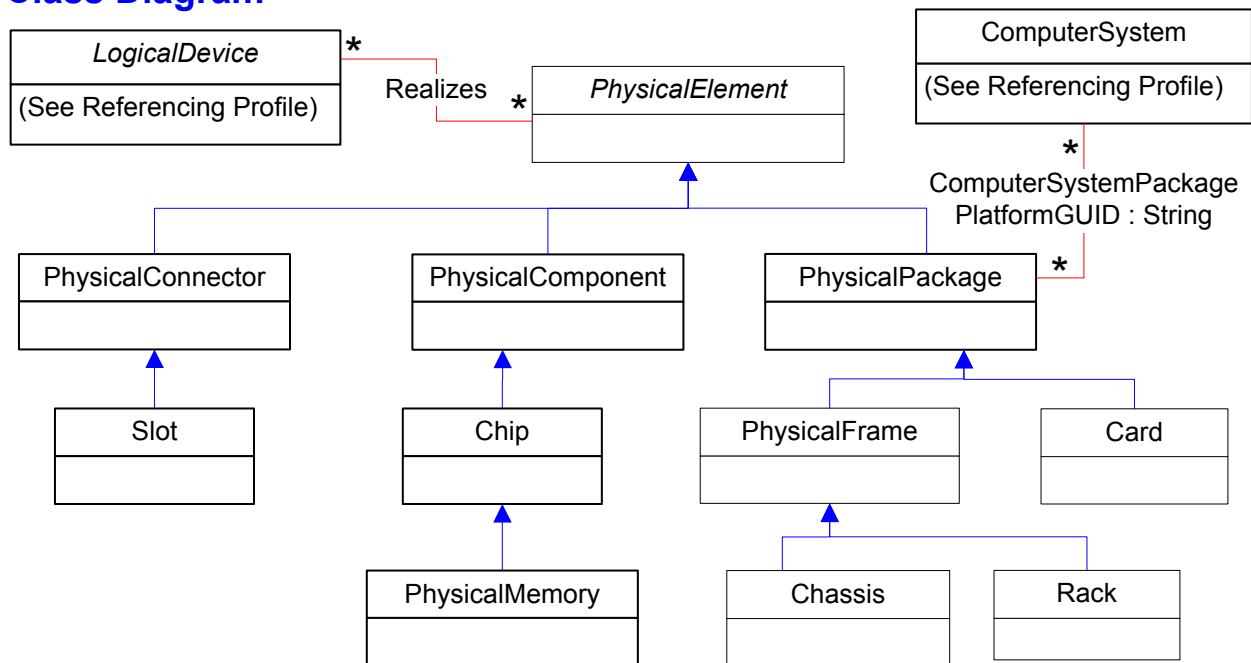
2461 10.1 Introduction

2462 The IPMI Entity Mapping extends the management capability of the IPMI to CIM mapping by
 2463 adding the capability to model the physical entities and corresponding logical devices
 2464 discoverable via the IPMI interface. FRU information, where available, will be represented in
 2465 the physical class being used to represent the physical entity. Logical devices are instantiated
 2466 where a corresponding physical class exists and can be identified from the entity type.

2467 The following class diagram describes the CIM classes involved in representing the physical
 2468 entities defined by the IPMI interface.

IPMI Physical Entities

Class Diagram



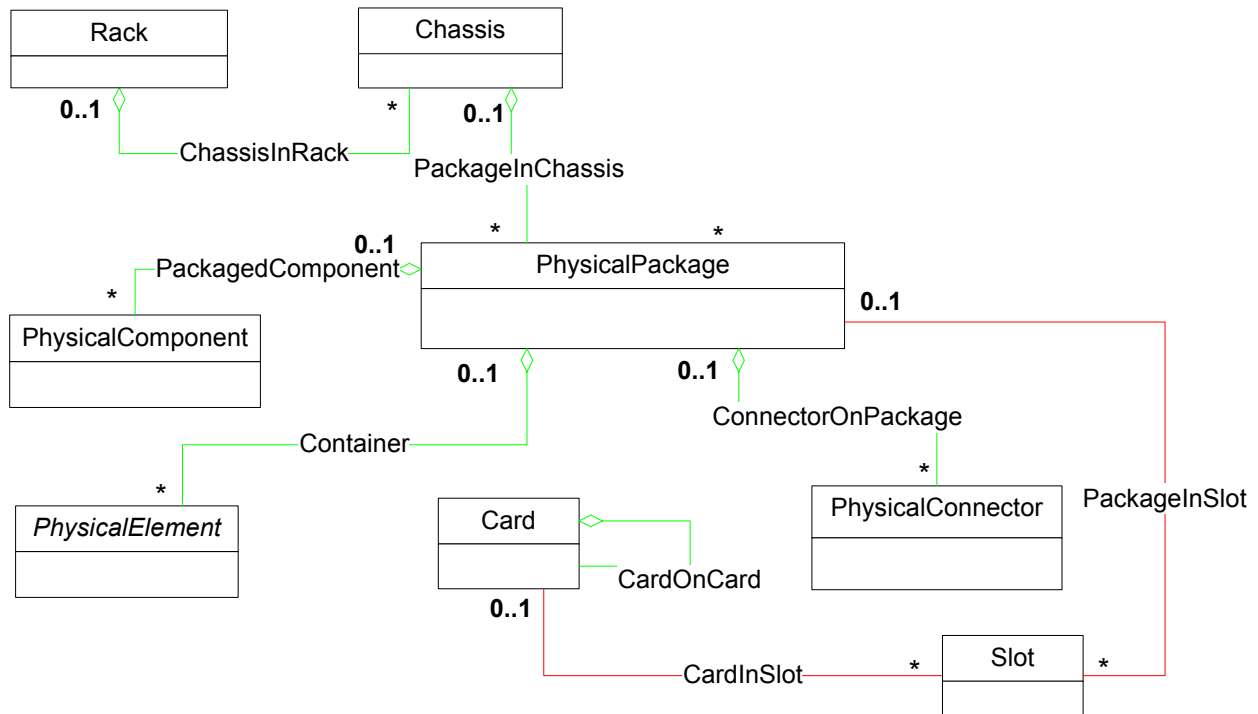
2469

2470 **Figure 13 Class Diagram – IPMI Physical Entities**

2471 The following class diagram describes the CIM associations involved in associating physical
 2472 classes according to entity in entity information from the IPMI interface.

Physical Entity Associations

Association Topology Diagram



2473

2474

Figure 14 Class Diagram – IPMI Physical Associations

2475

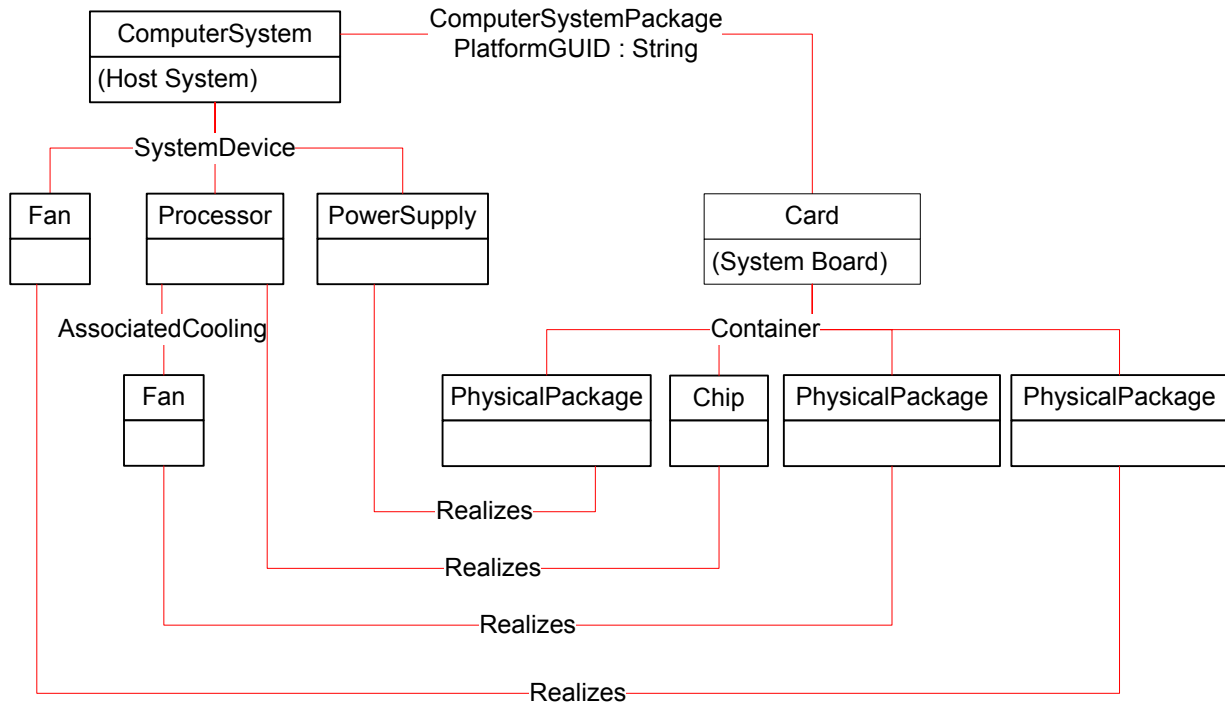
2476 10.2 Instance Diagrams

2477 10.2.1 Example: System Board as Physical Parent

2478 The following diagram describes a possible instantiation of physical and logical CIM classes
 2479 generated from entity information available via the IPMI interface. In this example, the IPMI
 2480 implementation has used the IPMI Entity “System Board” as the parent or top level container for
 2481 components in the system.

IPMI Entities

Instance Diagram (System Board as physical parent)



2482

2483

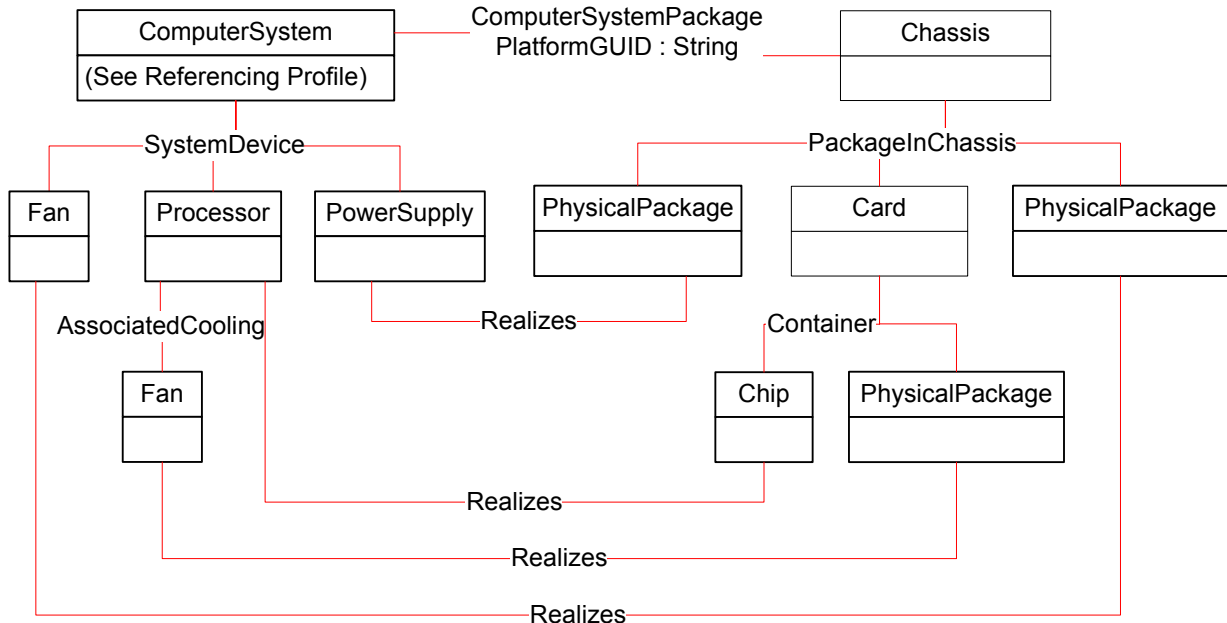
Figure 15 Instance Diagram – IPMI Entities (System Board Parent)

2484 10.2.2 Example: Chassis as Physical Parent

2485 The following diagram describes a possible instantiation of physical and logical CIM classes
 2486 generated from entity information available via the IPMI interface. In this example, the IPMI
 2487 implementation has used the IPMI Entity “System Chassis” as the parent or top level container
 2488 for components in the system.

IPMI Entities

Instance Diagram (Chassis as physical parent)



2489

2490

Figure 16 Instance Diagram – IPMI Entities (Chassis Parent)

2491

2492 10.3 Mapping Requirements

2493 Mapping IPMI entities and containers to CIM classes involves instantiating physical CIM classes
 2494 to model hierarchical relationships and to provide data structures for FRU information. In many
 2495 cases, CIM logical device classes also need to be instantiated to provide actual manageability of
 2496 the components being modeled in addition to providing representations that are meaningful to
 2497 consumers. Examples are the modeling of a Fan entity using both CIM_PhysicalPackage and
 2498 CIM_Fan classes for the case where both physical and logical are instantiated and the modeling
 2499 of a Disk Drive Bay where only the CIM_PhysicalPackage physical class is used.

2500

2501 10.3.1 Mapping Entity ID's to CIM Classes

2502 The following table defines the mapping from the IPMI Entity IDs (Table 37.13 IPMI 1.5
 2503 Specification) to physical and logical CIM classes. Note that some entity ID types do not have
 2504 corresponding logical device types in CIM. Additionally, the IPMI entity id's representing
 2505 software are mapped to CIM_SoftwareIdentity, which is not a physical class.

2506

IPMI Entity	CIM Physical Class	CIM Logical Device
Processor (03h)	CIM_Chip	CIM_Processor

IPMI Entity	CIM Physical Class	CIM Logical Device
Disk or disk bay (04h)	CIM_PhysicalPackage	CIM_DiskDrive
Peripheral bay (05h)	CIM_PhysicalPackage	
System management module (06h)	CIM_PhysicalPackage	CIM_ComputerSystem (BMC)
System board (07h)	CIM_Card	CIM_ComputerSystem (Main System)
Memory module (08h)	CIM_PhysicalPackage	CIM_Memory
Processor module (09h)	CIM_PhysicalPackage	CIM_Processor
Powey supply (0Ah)	CIM_PhysicalPackage	CIM_PowerSupply
Add-in card (0Bh)	CIM_Card	CIM_ComputerSystem
Front panel board (0Ch)	CIM_Card	
Back panel board (0Dh)	CIM_Card	
Power system board (0Eh)	CIM_Card	CIM_PowerSupply
Drive backplane (0Fh)	CIM_Card	
System internal expansion board (10h)	CIM_Card	CIM_ComputerSystem
Other system board (11h)	CIM_Card	
Processor board (12h)	CIM_Card	CIM_Processor
Power unit/power domain (13h)	CIM_PhysicalPackage	CIM_PowerSupply
Power module/DC-to-DC converter (14h)	CIM_PhysicalPackage	CIM_PowerSupply
Power management/power distribution board (15h)	CIM_Card	CIM_PowerSupply
Chassis back panel board (16h)	CIM_Card	
System chassis (17h)	CIM_Chassis	CIM_ComputerSystem (Managed System)
Sub-chassis (18h)	CIM_Chassis	
Other chassis board (19h)	CIM_Card	
Disk drive bay (1Ah)	CIM_PhysicalPackage	
Peripheral bay (1BH)	CIM_PhysicalPackage	
Device bay (1Ch)	CIM_PhysicalPackage	
Fan/cooling unit (1Eh)	CIM_PhysicalPackage	CIM_Fan
Cable/interconnect (1Fh)	CIM_PhysicalPackage	
Memory device (20h)	CIM_PhysicalPackage	CIM_Memory
System management software (21h)	CIM_SoftwareIdentity	
System firmware (22h)	CIM_SoftwareIdentity	
Operating system (23h)	CIM_SoftwareIdentity	
System bus (24h)	CIM_PhysicalPackage	
Group (25h)		
Remote management card (26h)	CIM_Card	CIM_ComputerSystem
External environment (27h)		
Battery (28h)	CIM_PhysicalPackage	CIM_Battery
Processing blade (29h)	CIM_PhysicalPackage	CIM_ComputerSystem
Connectivity switch (2Ah)	CIM_PhysicalPackage	CIM_ComputerSystem
Processor/memory module (2Bh)	CIM_PhysicalPackage	CIM_ComputerSystem
I/O module (2Ch)	CIM_PhysicalPackage	CIM_ComputerSystem

IPMI Entity	CIM Physical Class	CIM Logical Device
Processor/IO module (2Dh)	CIM_PhysicalPackage	CIM_ComputerSystem
Management controller firmware (2Eh)	CIM_SoftwareIdentity	
IPMI Channel (2Fh)		CIM_ServiceAccessPort

2507 **Table 15 IPMI Entity ID to CIM Class Mapping**

2508

2509 10.3.2 Entity Instantiation Algorithm

2510 The following algorithm is presented as the method for determining when to instantiate entities
 2511 as CIM classes. Use the Table 15 IPMI Entity ID to CIM Class Mapping to determine which
 2512 classes to instantiate according to entity ID.

2513 An “Entity” in IPMI is identified by the following pieces of information: The “Entity ID” which
 2514 identifies the type of the Entity the “Entity Instance”, which is a number that uniquely identifies
 2515 an instance of the entity - either relative to the overall system, or relative to a particular
 2516 management controller in the system.

2517 In the context of IPMI, the term “Entity” or “Entities” in this section shall be used as a shorthand
 2518 notation to refer to the full unique identification of a particular Entity. (That is, the combination
 2519 of Entity ID, Entity Instance, and, if the Entity Instance is device-relative, the identifying
 2520 information for the management controller associated that together uniquely identify an Entity or
 2521 Entities.)

2522 IPMI supports a type of Sensor Data Records called Entity Association Records. These records
 2523 provided a way of representing a hierarchical relationship among Entities by using linking a set
 2524 of ‘contained’ Entities under a single ‘container’ Entity.

2525 A given Entity is only allowed only be contained under a single, present, Entity Association
 2526 hierarchy at a time. I.e. it can’t appear in more than one existing Entity Association where the
 2527 top-level container entity is present. However, it’s possible that the SDRs could hold records for
 2528 two or more Entity Association hierarchies, where one hierarchy is present, and the others are
 2529 ‘absent’ because the container entity is absent (as indicated by a presence sensor or presence bit
 2530 associated with that container entity) . This is because some implementations may elect to
 2531 implement a set of SDRs that is a superset that covers multiple system configurations, and use
 2532 the Entity Association record mechanism as a way to organizing which sensors/Entities are
 2533 present for a given system configuration.

2534 Because of this, it is necessary to track which hierarchy a given Entity appears under. That way,
 2535 it allows an Entity to be ‘do not instantiate’ under one hierarchy, but can still exist to be
 2536 instantiated under another. This requires the following rule: if an Entity is listed under *any* Entity
 2537 Association hierarchy, then it must be part of a ‘present’ hierarchy in order to be instantiated.

2538 The following figure illustrates the case where a set of Entity Association records (left hand side
 2539 of figure) should be viewed as the equivalent of two separate Entity Association hierarchies
 2540 because two different container Entities are sharing common contained Entities. Note that this

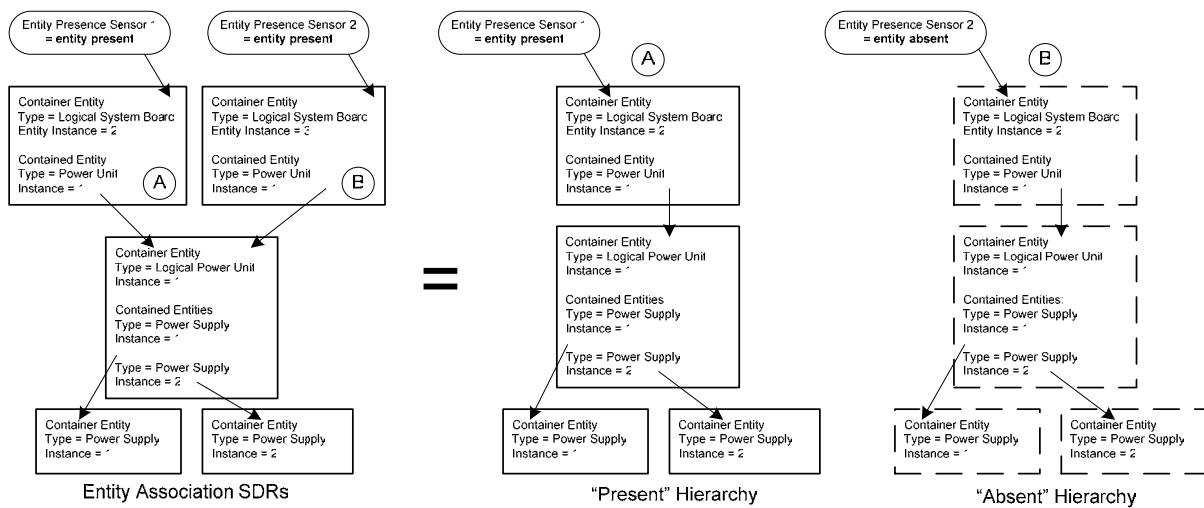
2541 requires one association to be ‘present’ and the other ‘absent’. In this case, this is accomplished
 2542 via an Entity Presence sensor that is associated with each ‘top’ container Entity.

2543 If the container Entity does not have any sensors associated with it, then the association is
 2544 assumed to be ‘present’.

2545 If the container Entity has one or more sensors associated with it, but all sensors are ‘scanning
 2546 disabled’ or are inaccessible, then the association should be ignored. I.e. software should act as if
 2547 that association was not even listed in the SDRs.

2548 If the container Entity has one or more ‘scanning enabled’ associated with it, and none of those
 2549 sensors are a presence sensor or presence bit that indicates that the container Entity is absent,
 2550 then the association is assumed to be ‘present’ (valid).

2551 If the container Entity has a presence sensor or bit associated with it that indicates the container
 2552 Entity is absent, then the overall association is said to be ‘*explicitly absent*’ and the container
 2553 Entity and any contained Entities and lower level associations should not be instantiated because
 2554 of being listed under that association. (Note that the same Entities may still become instantiated
 2555 because they’re listed under a different Entity association or association hierarchy that *is*
 2556 present.)



2557

2558 **Figure 17 “Present” and “Absent” Entity Association Hierarchies**

2559
 2560 1. First, determine which Type 01h and Type 02h SDRs can be ignored because their
 2561 management controller is not present. Once these SDRs have been identified, they do not
 2562 need to be re-examined in any further steps of this Entity instantiation process.

2563 ○ The BMC is always considered present and therefore does not require a device
 2564 locator record.

2565 ○ If the management controller associated with the sensor is not listed via a
 2566 Management Controller Device Locator Record (except for the BMC), assume

- 2567 any sensors associated with that controller are not present. This relationship can
2568 be identified by examining the fields for Device Slave Address and Channel
2569 Number listed in the Management Controller Device Locator record. These same
2570 values are used in the Type 01h and Type 02h sensor data records.
- 2571 ○ For each management controller listed, issue a *Get Device ID* command to verify
2572 that the controller is accessible. If the controller is not accessible, ignore any Type
2573 01h and Type 02h SDRs associated with sensors for that controller.
- 2574 2. Next, create a list of Entities that exist under Entity Associations, tagging each entry in
2575 the list with which association the Entity was found under. An Entity could possibly
2576 appear more than once in the list.
- 2577 ○ Start by finding the topmost ‘root’ container Entities. These are Entities that are
2578 not themselves listed as being contained Entities in any other Entity Association
2579 record. (The top of every Entity hierarchy must have a unique container Entity.
2580 Thus, the id information for the container Entity can be used as a way to identify a
2581 particular hierarchy)
- 2582 ○ Next, for each hierarchy, add the contained Entities to the list under that hierarchy,
2583 and then see if any of the contained Entities are container entities for any other
2584 entity association records. If so, add their contained entities to the list under the
2585 hierarchy and repeat the process until no more container entities are found under
2586 the hierarchy.
- 2587 ○ You now should have a list of the different Entity hierarchies (Entity
2588 Associations) and the Entities contained in each hierarchy. Note that at this point,
2589 a given Entity could appear in more than one hierarchy.
- 2590 3. Next, identify which Entities in the list should not be instantiated because they are
2591 contained under the hierarchy of a Device-relative Entity Association record where the
2592 management controller associated with the top of that hierarchy is not present.
- 2593 ○ For each Device-related Entity Association record, check to see whether the
2594 management controller for the record is present and accessible. If the management
2595 controller is not present and accessible, mark the contained Entities in the list as
2596 ‘do not instantiate’.
- 2597 ○ Check to see if any of the ‘do not instantiate’ Entities from the previous step are
2598 themselves container entities for either a Device-relative or regular (type 08h)
2599 Entity Association record under the hierarchy. If so, mark any subsequent
2600 contained Entities in that hierarchy as ‘do not instantiate’. Repeat until no more
2601 container entities are found that are listed in the ‘do not instantiate’ list.
- 2602 4. Next, identify which Entities should not be instantiated because they are contained under
2603 an Entity Association (either Device-related or regular [type 08h]) where the container
2604 Entity is ‘not present’ because an Entity Presence sensor indicates that the container
2605 Entity is not present.

2606 ○ Check the Type 01h and Type 02h records and see if there is an ‘Entity Presence’
 2607 sensor associated with the container Entity. If so, access the sensor to see if the
 2608 status indicates ‘Entity Absent’. If it does, mark the entity as ‘do not instantiate’
 2609 under the hierarchy.

2610 ○ If the Entity Presence sensor is disabled (not scanning) or inaccessible, assume
 2611 the Entity is not present and also mark the entity as ‘do not instantiate’ under the
 2612 hierarchy.

2613 At this point, any Entities in the hierarchy lists that are **not** marked as ‘do not instantiate’
 2614 are available to be instantiated. So you should go through the hierarchies and create two
 2615 lists of Entities: a ‘available for instantiation’ list of all the Entities the are are part of a
 2616 ‘present’ hierachy, and a ‘do not instantiate’ list, which is the list Entities that were found
 2617 in an Entity Association hierarchy and are not on the ‘avaiable for instantiation’ list.

2618 5. Identify whether additional Entities should be instantiated because they’re listed in Type
 2619 01h, 02h, or 03h SDRs.

2620 ○ For each Type 01h, Type 02h, or Type 03h SDR (that isn’t to be ignored because
 2621 of step ‘1’) see if the Entity associated with the sensor is on the ‘do not
 2622 instantiate’ list. If so, go on to the next SDR.

2623 ○ Otherwise, check to see if the sensor is of a type that includes a “Presence” bit
 2624 that indicates whether the Entity associated with the sensor is present or not. (This
 2625 only applies to Type 01h or Type 02h SDRs that support readable sensors). If the
 2626 bit indicates the Entity is absent, add that Entity to the ‘do not instantiate’ list and
 2627 remove it from ‘available from instantiation’ list if it’s there . Otherwise, add the
 2628 Entity to the ‘available for instantiation’ list.

2629 6. Certain types of sensors imply an Entity that can be instantiated under CIM. For example,
 2630 Fan and Processor sensors imply the existence of Fan and Processor entities. Identify
 2631 Entities that should be instantiated based on the sensor type.

2632 ○ For each Type 01h, Type 02h, or Type 03h SDR (that isn’t to be ignored because
 2633 of step ‘1’) see if the Entity associated with the sensor is on the ‘do not
 2634 instantiate’ list. If so, go on to the next SDR.

2635 ○ If the Sensor Type (Sensor/Event Type) in the SDR does not match the Entity ID
 2636 given in the SDR, then a separate Entity based on the sensor type should be added
 2637 to the ‘available for instantiation’ list, provided that the Sensor/Event Type is one
 2638 of the types in the following table. Since there is not an explicit Entity Instance
 2639 number available in this case, an Entity Instance number must be synthesized. The
 2640 Entity Instance number should recored as the combination of Sensor Number
 2641 (SDR type 1|2 table byte 8), Owner LUN (SDR type 1|2 table byte 7), Owner ID
 2642 (SDR type 1|2 table byte 6) so that the instance ID can ultimately be formulated as
 2643 a CIM_Sensor.DeviceID [See section 3.4.2.4, CIM_Sensor.DeviceID].

2644

Table 16 “Physical” Sensor Types to Entity Mapping

Sensor Type	Sensor Type Code	Entity
Fan	04h	Fan (29)
Physical Security (Chassis Intrusion)	05h	System Chassis (23)
Processor	07h	Processor (3)
Power Supply (also used for power converters [e.g. DC-to-DC converters] and VRMs [voltage regulator modules]).	08h	Power Supply (10)
Power Unit	09h	Power Unit (19)
Cooling Device	0Ah	Cooling Device (29)
Button / Switch	14h	Sub-Chassis (18h)
Module / Board	15h	Other Chassis Board (19)
Microcontroller / Coprocessor	16h	Processor (3)
Add-in Card	17h	Add-In Card (11)
Chassis	18h	System Chassis (23)
Chip Set	19h	Processor (3)
Other FRU	1Ah	???
Cable / Interconnect	1Bh	Cable/Interconnect (31)
Terminator	1Ch	Cable/Interconnect (31)
Slot / Connector	21h	Cable/Interconnect (31)
Watchdog 2	23h	CIM_WatchDog
Monitor ASIC / IC	26h	CIM_Chip
LAN	27h	Remote Management Communication Device (38)
Battery	29h	Battery (28)
Session Audit	2Ah	???

2645
2646

7. Add Entities to instantiate because they’re listed in FRU Device Locator records.

2647
2648
2649

- For each FRU Device Location record, check to see if the Entity is already listed in the ‘available for instantiation’ list. If not, add the Entity to the ‘available for instantiation’ list if the corresponding FRU Device is accessible.

2650
2651

8. Add Entities to instantiate because they’re listed in Management Controller Device Locator records.

2652
2653
2654
2655
2656
2657
2658
2659

- Primary FRU Devices are FRU devices that are implemented as FRU Device 000b on LUN 00b of a management controller. These devices do not have FRU Device Locator records. Instead, the presence of the FRU Device and information on the Entity associated with the FRU Device is provided by the Management Controller Device Locator record associated with the controller. For each Management Device Locator record, check to see if the Entity is already listed in the ‘available for instantiation’ list. If not, add the Entity to the ‘available for instantiation’ list if the corresponding FRU Device is accessible.

2660
2661

9. Add Entities to instantiate because they’re implied by FRU Data. Refer to the following table.

2662

Table 17 FRU - Entity Instantiation

FRU Device is:	Entity Type is:	FRU Device Contains:	FRU Data is for:
BMC Primary FRU	n/a	Chassis Info Area + Product Info	Assume Chassis Info is for overall

Device (FRU Device 000b on BMC LUN 00b)		Area	system chassis and Product Info is for overall computer system as a product, and that the Chassis Info is for instance 1 of the system chassis (Entity Type 11h) unless that Entity has already been explicitly identified by a FRU Device Locator or Management Controller Device Locator record. In that case, instantiate a unique Entity for a different instance of 'system chassis'.
		Product Info Area but no Chassis Info Area	Assume Product Info is for overall System UNLESS another FRU device is found that contains <i>both</i> a Chassis Info and a Product Info area and is associated with instance 1 of the system chassis (Entity Type 11h)
		Chassis Info Area but no Product Info Area	Assume Chassis Info is for overall System UNLESS another FRU device is found that contains <i>both</i> a Chassis Info and a Product Info area and is associated with instance 1 of the system chassis (Entity Type 11h).
		Board Info Area	Assume Board Info is for instance 1 of the main system board (Entity Type 7h) unless that Entity has been explicitly identified by a FRU Device Locator or Management Controller Device Locator record. In that case, instantiate a unique Entity for an 'other system board (11h)'.
Primary FRU Device on Satellite controller (Device 000b on LUN 00b)	Specified by Management Controller Device Locator Record	Board Info Area, Chassis Info Area, or Product Info area	FRU data is associated with type of Entity given by the Entity ID information in the Management Controller Device Locator record. If Chassis info is present, & there is no other Chassis info is present, assume info is for overall system chassis and Product Info, if also present, is for overall computer system as a product, and that the Chassis Info is for instance 1 of the system chassis (Entity Type 11h) unless that Entity has already been explicitly identified by a FRU Device Locator or Management Controller Device Locator record. In that case, instantiate a unique Entity for a different instance of 'system chassis'.
	Unspecified	Chassis Info Area + Product Info Area	If BMC Primary FRU Device does not already have Chassis Info and Product Info Areas, and a FRU area Assume Chassis Info is for overall system chassis and Product Info is for overall computer system as a product, and that the Chassis Info is for instance 1 of the system chassis (Entity Type 11h) unless that Entity has already been explicitly identified by a FRU Device Locator or

			Management Controller Device Locator record. Otherwise, instantiate a unique Entity for a different instance of 'system chassis' associated with the management controller.
		Product Info Area but no Chassis Info Area	Assume Product Info is for overall System UNLESS there is Product Info in the BMC Primary FRU Device, or another FRU device is found that contains <i>both</i> a Chassis Info and a Product Info area and is associated with instance 1 of the system chassis (Entity Type 11h)
		Chassis Info Area but no Product Info Area	Assume Chassis Info is for overall System UNLESS there is Chassis Info in the BMC Primary FRU Device, or another FRU device is found that contains <i>both</i> a Chassis Info and a Product Info area and is associated with instance 1 of the system chassis (Entity Type 11h)
		Board Info Area	Assume Board Info is for instance 1 of the main system board (Entity Type 7h) unless that Entity has been explicitly identified by a FRU Device Locator or Management Controller Device Locator record. In that case, instantiate a unique Entity for an 'other system board (11h)'.
Identified by FRU Device Locator Record	Specified by FRU Device Locator Record	Board Info Area, Chassis Info Area, or Product Info area	FRU data is associated with type of Entity given by the Entity ID information in the FRU Device Locator record. Any combination of Product, Chassis, and Board Info areas may be present. Appropriate CIM "Package" classes should be instantiated for each type of record. The Entity itself should be added to the 'available for instantiation' if not already present.
	Unspecified	Chassis Info Area + Product Info Area	If Chassis Info area is present, and a Chassis Info area is <i>not</i> provided in the Primary FRU Device of the BMC, assume Chassis Info is for overall system chassis and Product Info is for overall computer system as a product. Assume Chassis Info is for instance 1 of the system chassis (Entity Type 11h) unless that Entity has already been explicitly identified by a FRU Device Locator or Management Controller Device Locator record. In that case, instantiate a unique Entity for a different instance of 'system chassis'.
		Product Info Area but no Chassis Info Area	Assume Product Info is for overall System UNLESS another FRU device is found that contains <i>both</i> a Chassis Info and a Product Info area.
		Chassis Info Area but no Product Info Area	Assume Chassis Info is for overall System UNLESS another FRU device is found that contains <i>both</i> a Chassis Info and a Product Info area and is

			either associated with instance 1 of the system chassis (Entity Type 11h), or is in the Primary FRU Device of the BMC.
		Board Info Area	Board Info is for FRU that the management controller is located on.

2663 **10.3.3 Entity Instance Correlation**

2664 Instrumentation **MUST** keep track of the IPMI Entity Instance identifiers and correlate during
 2665 SDR discovery so as to only produce one set of CIM instances representing a particular IPMI
 2666 entity.

2667 **10.4 CIM_PhysicalPackage Mapping**

2668 **10.4.1 CIM_PhysicalPackage Methods**

2669 None.

2670 **10.4.2 CIM_PhysicalPackage Properties**

2671 **REQUIRED** properties are properties that **MUST** be present. **RECOMMENDED** properties are
 2672 properties that **MAY** be present. For example, there may be properties that are only filled out in
 2673 special configurations of the CIM Schema.

2674 Several property vales are dependent on the availability of associated FRU information. The
 2675 Product Info Area as defined in Platform Management FRU Information Storage Definition v1.1
 2676 is mapped for this class.

2677 Properties that are listed in the Required section in the following table **MUST** be implemented as
 2678 defined in the subsections following this table unless the Notes column for each property
 2679 specifies the defining reference.

2680 **Table 18 CIM_PhysicalPackage Class Properties**

Required Properties	Notes
Tag	Key
CreationClassName	Key
PackageType	
VendorEquipmentType	
HealthState	
OperationalStatus	
Description	
ElementName	
<i>Recommended Properties</i>	Notes
VendorCompatibilityString(E)	

2681

Recommended Properties	Notes
PoweredOn	
CanBeFRUed	
Manufacturer	If FRU available See 10.4.2.4
Model	If FRU available See 10.4.2.5
SKU	
OtherPackageType	
RemovalConditions	
ManufactureDate	
ActiveCooling	
Version	
SerialNumber	If FRU available See 10.4.2.7
PartNumber	If FRU available See 10.4.2.6
Height	
Depth	
Width	
Weight	
IdentifyingDescriptions	
OtherIdentifyingInfo	
UserTracking	
StatusDescriptions	
InstallDate	
Name	If FRU available See 10.4.2.4
Caption	

2682

2683 The following subsections describe procedures for generating values for various properties of
2684 CIM classes required in this profile.

2685 **10.4.2.1 CIM_PhysicalPackage.Tag**

2686 An arbitrary string that uniquely identifies the Physical Element and serves as the Element ' s
2687 key.

2688 Formulate as string:

2689 (string)<Entity ID from SDR> || (string)<Entity Instance from SDR>

2690 || (string)<Sensor Number from SDR>

2691 **10.4.2.2 CIM_PhysicalPackage.CreationClassName**

2692 Per the DMTF definition of the class property: The name of the class or the subclass used in the
2693 creation of an instance. This will typically be “CIM_PhysicalPackage”.

2694 Formulate as a string:

2695 <"CIM_PhysicalPackage" or the name of the subclass created>

2696 **10.4.2.3 CIM_PhysicalPackage.PackageType**

2697 Values for CIM_PhysicalPackage.PackageType are mapped to values of Entity ID from the

2698 SDR.

2699 Formulate as integer:

2700 if Entity ID = 04h (disk or disk bay)

2701 CIM_PhysicalPackage.PackageType = 15 (Storage Media Package)

2702 elseif EntityID = 05h (peripheral bay)

2703 CIM_PhysicalPackage.PackageType = 5 (Container/Frame Slot)

2704 elseif EntityID = 06h (system management module)

2705 CIM_PhysicalPackage.PackageType = 1 (Other)

2706 CIM_PhysicalPackage.OtherPackageType = "System Management Module"

2707 elseif EntityID = 08h (memory module)

2708 CIM_PhysicalPackage.PackageType = 13 (Memory)

2709 elseif EntityID = 09h (processor module)

2710 CIM_PhysicalPackage.PackageType = 12 (Processor)

2711 elseif EntityID = 0Ah (power supply)

2712 CIM_PhysicalPackage.PackageType = 6 (Power Supply)

2713 elseif EntityID = 13h (power unit/power domain)

2714 CIM_PhysicalPackage.PackageType = 1 (Other)

2715 CIM_PhysicalPackage.OtherPackageType = "Power unit/Power domain"

2716 elseif EntityID = 14h (power module/DC-to-DC converter)

2717 CIM_PhysicalPackage.PackageType = 1 (Other)

2718 CIM_PhysicalPackage.OtherPackageType = "Power module/DC-to-DC converter"

2719 elseif EntityID = 1Ah (disk drive bay)

2720 CIM_PhysicalPackage.PackageType = 1 (Other)

2721 CIM_PhysicalPackage.OtherPackageType = “Disk drive bay”

2722 elseif EntityID = 1Bh (peripheral bay)

2723 CIM_PhysicalPackage.PackageType = 1 (Other)

2724 CIM_PhysicalPackage.OtherPackageType = “Peripheral bay”

2725 elseif EntityID = 1Ch (device bay)

2726 CIM_PhysicalPackage.PackageType = 1 (Other)

2727 CIM_PhysicalPackage.OtherPackageType = “Device bay”

2728 elseif EntityID = 1Eh (fan/cooling unit)

2729 CIM_PhysicalPackage.PackageType = 7 (Fan)

2730 elseif EntityID = 1Fh (cable/interconnect)

2731 CIM_PhysicalPackage.PackageType = 1 (Other)

2732 CIM_PhysicalPackage.OtherPackageType = “Cable/Interconnect”

2733 ElseIf EntityID = 20h (memory device)

2734 CIM_PhysicalPackage.PackageType = 13 (Memory)

2735 ElseIf EntityID = 24h (system bus)

2736 CIM_PhysicalPackage.PackageType = 4 (Cross Connect/Backplane)

2737 elseif EntityID = 28h (battery)

2738 CIM_PhysicalPackage.PackageType = 1 (Other)

2739 CIM_PhysicalPackage.OtherPackageType = “Battery”

2740 elseif EntityID = 29h (processing blade)

2741 CIM_PhysicalPackage.PackageType = 1 (Other)

2742 CIM_PhysicalPackage.OtherPackageType = “Processing Blade”

2743 elseif EntityID = 2Ah (connectivity switch)

2744 CIM_PhysicalPackage.PackageType = 1 (Other)

2745 CIM_PhysicalPackage.OtherPackageType = “Connectivity Switch”

2746 elseif EntityID = 2Bh (processor/memory module)

2747 CIM_PhysicalPackage.PackageType = 1 (Other)

2748 CIM_PhysicalPackage.OtherPackageType = “Processor/Memory Module”

2749 elseif EntityID = 2Ch (I/O module)

2750 CIM_PhysicalPackage.PackageType = 1 (Other)

2751 CIM_PhysicalPackage.OtherPackageType = “I/O Module”

2752 elseif EntityID = 2Dh (processor/IO module)

2753 CIM_PhysicalPackage.PackageType = 1 (Other)

2754 CIM_PhysicalPackage.OtherPackageType = “Processor/IO Module”

2755 **10.4.2.4 CIM_PhysicalPackage.Manufacturer**

2756 Formulate as a string:

2757 (string) associated FRU 1.1 Product Info Area – Manufacturer Name field

2758 **10.4.2.5 CIM_PhysicalPackage.Model**

2759 Formulate as a string:

2760 (string) associated FRU 1.1 Product Info Area – Product Part/Model Number field

2761 **10.4.2.6 CIM_PhysicalPackage.PartNumber**

2762 Formulate as a string:

2763 (string) associated FRU 1.1 Product Info Area – Product Part/Model Number field

2764 **10.4.2.7 CIM_PhysicalPackage.SerialNumber**

2765 Formulate as a string:

2766 (string) associated FRU 1.1 Product Info Area – Product Serial Number field

2767 **10.4.2.8 CIM_PhysicalPackage.Name**

2768 Formulate as a string:

2769 (string) associated FRU 1.1 Product Info Area – Product Name field

2770 10.4.2.9 CIM_PhysicalPackage.HealthState**2771 10.4.2.10 CIM_PhysicalPackage.OperationalStatus****2772 10.4.2.11 CIM_PhysicalPackage.Description**

2773 Formulate as string:

2774 (string)< CIM_PhysicalPackage.PackageType or
2775 CIM_PhysicalPackage.OtherPackageType > || “(” || (string)<Entity ID from SDR> || “.” ||
2776 (string)<Entity Instance from SDR>

2777 10.4.2.12 CIM_PhysicalPackage.ElementName

2778 Formulate as string:

2779 (string)< CIM_PhysicalPackage.PackageType or
2780 CIM_PhysicalPackage.OtherPackageType > || “(” || (string)<Entity ID from SDR> || “.” ||
2781 (string)<Entity Instance from SDR>

2782 10.5 CIM_Card Mapping**2783 10.5.1 CIM_Card Methods**

2784 None.

2785 10.5.2 CIM_Card Properties

2786 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
2787 properties that MAY be present. For example, there may be properties that are only filled out in
2788 special configurations of the CIM Schema.

2789 Several property vales are dependent on the availability of associated FRU information. The
2790 Board Info Area as defined in Platform Management FRU Information Storage Definition v1.1 is
2791 mapped for this class.

2792 Properties that are listed in the Required section in the following table MUST be implemented as
2793 defined in the subsections following this table unless the Notes column for each property
2794 specifies the defining reference.

2795

Table 19 CIM_Card Class Properties

Required Properties	Notes
Tag	Key
CreationClassName	Key
HostingBoard	
PackageType	
OtherPackageType	
HealthState	
OperationalStatus	
Description	
ElementName	
Recommended Properties	Notes
Manufacturer	If FRU available See 10.5.2.5
Model	
SKU	
VendorEquipmentType	
VendorCompatibilityString(E)	
PoweredOn	
CanBeFRUed	
SlotLayout	
RequiresDaughterBoard	
SpecialRequirements	
RequirementsDescription	
OperatingVoltages	
RemovalConditions	
ManufactureDate	If FRU available See 10.5.2.6
ActiveCooling	
Version	
SerialNumber	If FRU available See 10.5.2.8
PartNumber	If FRU available See 10.5.2.7
Height	
Depth	
Width	
Weight	
IdentifyingDescriptions	
OtherIdentifyingInfo	
UserTracking	
StatusDescriptions	
InstallDate	
Name	If FRU available See 10.5.2.9

2796

<i>Recommended Properties</i>	Notes
Caption	

2797

2798 The following subsections describe procedures for generating values for various properties of
2799 CIM classes required in this profile.

2800 **10.5.2.1 CIM_Card.Tag**

2801 An arbitrary string that uniquely identifies the Physical Element and serves as the Element 's
2802 key. Formulate as string:

2803 (string)<Entity ID from SDR> || “:” || (string)<Entity Instance from SDR> || “:” ||

2804 || (string)<Sensor Number from SDR>

2805 **10.5.2.2 CIM_Card.CreationClassName**

2806 Per the DMTF definition of the class property: The name of the class or the subclass used in the
2807 creation of an instance. This will typically be “CIM_Card”.

2808 Formulate as a string:

2809 <”CIM_Card” or the name of the subclass created>

2810 **10.5.2.3 CIM_Card.HostingBoard**

2811 Per the MOF, this is supposed to be a Boolean indicating whether or not the card is a Mother
2812 board.

2813 Formulate as string:

2814 if Entity ID = 07h (System board)

2815 CIM_Card.HostingBoard = “System board”

2816 else

2817 CIM_Card.HostingBoard = NULL

2818 **10.5.2.4 CIM_Card.PackageType**

2819 Values for CIM_Card.PackageType are mapped to values of Entity ID from the SDR.

2820 Formulate as integer:

2821 if Entity ID = 07h (System board)

2822 CIM_Card.PackageType = 1 (Other)

```
2823         CIM_Card.OtherPackageType = "System board"
2824 elseif EntityID = 0Bh (Add-in card)
2825         CIM_Card.PackageType = 1 (Other)
2826         CIM_Card.OtherPackageType = "Add-in card"
2827 elseif EntityID = 0Ch (Front panel board)
2828         CIM_Card.PackageType = 1 (Other)
2829         CIM_Card.OtherPackageType = "Front panel board"
2830 elseif EntityID = 0Dh (Back panel board)
2831         CIM_Card.PackageType = 1 (Other)
2832         CIM_Card.OtherPackageType = "Back panel board"
2833 elseif EntityID = 0Eh (Power system board)
2834         CIM_Card.PackageType = 1 (Other)
2835         CIM_Card.OtherPackageType = "Power system board"
2836 elseif EntityID = 0Fh (Drive backplane)
2837         CIM_Card.PackageType = 1 (Other)
2838         CIM_Card.OtherPackageType = "Drive backplane"
2839 elseif EntityID = 10h (System internal expansion backplane)
2840         CIM_Card.PackageType = 1 (Other)
2841         CIM_Card.OtherPackageType = "System internal expansion backplane"
2842 elseif EntityID = 11h (Other system board)
2843         CIM_Card.PackageType = 1 (Other)
2844         CIM_Card.OtherPackageType = "Other system board"
2845 elseif EntityID = 12h (Processor board)
2846         CIM_Card.PackageType = 1 (Other)
2847         CIM_Card.OtherPackageType = "Processor board"
2848 elseif EntityID = 15h (Power management/power distribution board)
```

2849 CIM_Card.PackageType = 1 (Other)
2850 CIM_Card.OtherPackageType = “Power management/power distribution board”
2851 elseif EntityID = 16h (Chassis back panel board)
2852 CIM_Card.PackageType = 1 (Other)
2853 CIM_Card.OtherPackageType = “Chassis back panel board”
2854 elseif EntityID = 19h (Other chassis board)
2855 CIM_Card.PackageType = 1 (Other)
2856 CIM_Card.OtherPackageType = “Other chassis board”
2857 elseif EntityID = 26h (Remote management card)
2858 CIM_Card.PackageType = 1 (Other)
2859 CIM_Card.OtherPackageType = “Remote management card”

2860 **10.5.2.5 CIM_Card.Manufacturer**

2861 Formulate as a string:

2862 (string) associated FRU 1.1 Board Info Area – Board Manufacturer Name field

2863 **10.5.2.6 CIM_Card.ManufacturerDate**

2864 Formulate as a string:

2865 (CIM datetime data type) associated FRU 1.1 Board Info Area – Mfg. Date/Time field

2866 **10.5.2.7 CIM_Card.PartNumber**

2867 Formulate as a string:

2868 (string) associated FRU 1.1 Board Info Area – Board Part/Model Number field

2869 **10.5.2.8 CIM_Card.SerialNumber**

2870 Formulate as a string:

2871 (string) associated FRU 1.1 Board Info Area – Board Serial Number field

2872 **10.5.2.9 CIM_Card.Name**

2873 Formulate as a string:

2874 (string) associated FRU 1.1 Board Info Area – Board Product Name field

2875 **10.5.2.10 CIM_Card.HealthState**

2876 **10.5.2.11 CIM_Card.OperationalStatus**

2877 **10.5.2.12 CIM_Card.Description**

2878 Formulate as string:

2879 (string)< CIM_Card.PackageType or CIM_Card.OtherPackageType > || “(” ||
 2880 (string)(SensorNumber) || “.” || (string)(OwnerLUN) || “.” || string(OwnerID) || “): ”
 2881 ||(string)(EStr(<CIM_Sensor.SensorType>)) || “ for ” || (string)(<IPMI_SDR_EntityID>)
 2882 || “ “ || (string)(<IPMI_SDR_EntityInstanceNumber>)

2883 **10.5.2.13 CIM_Card.ElementName**

2884 Formulate as string:

2885 (string)< >< CIM_Card.PackageType or CIM_Card.OtherPackageType > || “(” ||
 2886 (string)<Entity ID from SDR> || “:” || (string)<Entity Instance from SDR> || “)”

2887 **10.6 CIM_Chip Mapping**

2888 **10.6.1 CIM_Chip Methods**

2889 None.

2890 **10.6.2 CIM_Chip Properties**

2891 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
 2892 properties that MAY be present. For example, there may be properties that are only filled out in
 2893 special configurations of the CIM Schema.

2894 Properties that are listed in the Required section in the following table MUST be
 2895 implemented as defined in the subsections following this table unless the Notes column for each
 2896 property specifies the defining reference.

2897

Table 20 CIM_Chip Class Properties

Required Properties	Notes
Tag	Key
CreationClassName	Key
HealthState	
OperationalStatus	
Description	
ElementName	
Recommended Properties	Notes
Manufacturer	
Model	
Version	
VendorEquipmentType	
PoweredOn	
CanBeFRUed	
SKU	
RemovalConditions	
ManufactureDate	
Removable	
Replaceable	
HotSwappable	
UserTracking	
Version	
SerialNumber	
PartNumber	
FormFactor	
StatusDescriptions	
InstallDate	
Name	
Caption	

2898

2899

2900 The following subsections describe procedures for generating values for various properties of
 2901 CIM classes required in this profile.

2902 **10.6.2.1 CIM_Chip.Tag**

2903 An arbitrary string that uniquely identifies the Physical Element and serves as the Element 's
 2904 key. Formulate as string:

2905 (string)<Entity ID from SDR> || “:” || (string)<Entity Instance from SDR> || “:” ||

2906 || (string)<Sensor Number from SDR>

2907 **10.6.2.2 CIM_Chip.CreationClassName**

2908 Per the DMTF definition of the class property: The name of the class or the subclass used in the
2909 creation of an instance. This will typically be “CIM_Chip”.

2910 Formulate as a string:

2911 <”CIM_Chip” or the name of the subclass created>

2912 **10.6.2.3 CIM_Chip.HealthState**

2913 **10.6.2.4 CIM_Chip.OperationalStatus**

2914 **10.6.2.5 CIM_Chip.Description**

2915 Formulate as string:

2916 (string)< SDR entry bytes 49+, see IPMI v1.5 Table 37-1> || “(” ||
2917 (string)(SensorNumber) || “.” || (string)(OwnerLUN) || “.” || string)(OwnerID) || “): ”
2918 ||(string)(EStr(<CIM_Sensor.SensorType>)) || “ for ” || (string)(<IPMI_SDR_EntityID>)
2919 || “ “ || (string)(<IPMI_SDR_EntityInstanceNumber>)

2920 **10.6.2.6 CIM_Chip.ElementName**

2921 Formulate as string:

2922 (string)< SDR entry bytes 49+, see IPMI v1.5 Table 37-1> || “(” || (string)<Entity ID from SDR>
2923 || “:” || (string)<Entity Instance from SDR> || “)”

2924 **10.7 CIM_Chassis Mapping**

2925 **10.7.1 CIM_Chassis Methods**

2926 None Defined.

2927 **10.7.2 CIM_Chassis Properties**

2928 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
2929 properties that MAY be present. For example, there may be properties that are only filled out in
2930 special configurations of the CIM Schema.

2931 Several property vales are dependent on the availability of associated FRU information. The
2932 Chassis Info Area as defined in Platform Management FRU Information Storage Definition v1.1
2933 is mapped for this class.

2934 Properties that are listed in the Required section in the following table MUST be implemented as
 2935 defined in the subsections following this table unless the Notes column for each property
 2936 specifies the defining reference.

2937 **Table 21 CIM_Chassis Class Properties**

Required Properties	Notes
Tag	Key
CreationClassName	Key
ChassisPackageType	
HealthState	
OperationalStatus	
Description	
ElementName	
<i>Recommended Properties</i>	Notes
Manufacturer	
Model	
SKU	
VendorEquipmentType	
VendorCompatibilityString(E)	
PoweredOn	
CanBeFRUed	
LockPresent	
NumberOfPowerCords	
CurrentRequiredOrProduced	
HeatGeneration	
ChassisTypeDescription	
MultipleSystemSupport	
RackMountable	
CableManagementStrategy	
ServicePhilosophy	
ServiceDescriptions	
AudibleAlarm	
VisibleAlarm	
SecurityBreach	
BreachDescription	
IsLocked	
OtherPackageType	
RemovalConditions	
ManufactureDate	
ActiveCooling	

Recommended Properties	Notes
Version	
SerialNumber	If FRU available See Section 10.7.2.5
PartNumber	If FRU available See Section 10.7.2.4
Height	
Depth	
Width	
Weight	
IdentifyingDescriptions	
OtherIdentifyingInfo	
UserTracking	
StatusDescriptions	
InstallDate	
Name	
Caption	

2939 **10.7.2.1 CIM_Chassis.Tag**

2940 An arbitrary string that uniquely identifies the Physical Element and serves as the Element ' s
2941 key. Formulate as a string:

2942
2943 (string)<Entity ID from SDR> || “:” || (string)<Entity Instance from SDR> || “:” ||
2944 || (string)<Sensor Number from SDR>

2945 **10.7.2.2 CIM_Chassis.CreationClassName**

2946 Per the DMTF definition of the class property: The name of the class or the subclass used in the
2947 creation of an instance. This will typically be “CIM_Chassis”.

2948 Formulate as a string:

2949 <”CIM_Chassis” or the name of the subclass created>

2950

2951 **10.7.2.3 CIM_Chassis.ChassisPackageType**

2952 Values for CIM_Chassis.ChassisPackageType are mapped to values of Entity ID from the SDR.

2953 Formulate as integer:

2954 if Entity ID = 17h (System chassis)

2955 CIM_Chassis.ChassisPackageType = 17 (Main System Chassis)

2956 elseif EntityID = 18h (Sub-chassis)

- 2957 CIM_Chassis.ChassisPackageType = 19 (SubChassis)
- 2958 **10.7.2.4 CIM_Chassis.PartNumber**
2959 Formulate as a string:
- 2960 (string) associated FRU 1.1 Chassis Info Area – Chassis Part Number field
- 2961 **10.7.2.5 CIM_Chassis.SerialNumber**
2962 Formulate as a string:
- 2963 (string) associated FRU 1.1 Chassis Info Area – Chassis Serial Number field
- 2964 **10.7.2.6 CIM_Chassis.HealthState**
- 2965 **10.7.2.7 CIM_Chassis.OperationalStatus**
- 2966 **10.7.2.8 CIM_Chassis.Description**
2967 Formulate as string:
- 2968 (string)< CIM_Chassis.ChassisPackageType > || “(” || (string)(SensorNumber) || “.” ||
2969 (string)(OwnerLUN) || “.” || string)(OwnerID) || “): ”
2970 ||(string)(EStr(<CIM_Sensor.SensorType>)) || “ for ” || (string)(<IPMI_SDR_EntityID>)
2971 || “ “ || (string)(<IPMI_SDR_EntityInstanceNumber>)
- 2972 **10.7.2.9 CIM_Chassis.ElementName**
2973 Formulate as string:
- 2974 (string)<CIM_Chassis.ChassisPackageType> || “(” || (string)<Entity ID from SDR> || “:”
2975 || (string)<Entity Instance from SDR> || “)”
- 2976
- 2977 **10.8 CIM_Processor Mapping**
- 2978 **10.8.1 CIM_Processor Methods**
2979 No Mapping Defined.
- 2980 **10.8.2 CIM_Processor Properties**
2981 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
2982 properties that MAY be present. For example, there may be properties that are only filled out in
2983 special configurations of the CIM Schema.

2984 Properties that are listed in the Required section in the following table MUST be implemented as
 2985 defined in the subsections following this table unless the Notes column for each property
 2986 specifies the defining reference.

2987 **Table 22 CIM_Processor Class Properties**

Required Properties	Notes
SystemCreationClassName	Key
SystemName	Key
CreationClassName	Key
DeviceID	Key
UniqueID	
EnabledState	
HealthState	
OperationalStatus	
Description	
ElementName	
<i>Recommended Properties</i>	<i>Notes</i>
Family	
CurrentClockSpeed	
MaxClockSpeed	
ExternalClockSpeed	
Role	
OtherFamilyDescription	
DataWidth	
AddressWidth	
Stepping	
CPUStatus	
OtherEnabledState	
StatusDescriptions	
Name	
Caption	
Description	
UpgradeMethod	
LoadPercentage	
IdentifyingDescriptions	
AdditionalAvailability	
OtherIdentifyingInfo	
TimeOfLastChange	
InstallDate	

2988

<i>Recommended Properties</i>	<i>Notes</i>
ElementName	

2989 **10.8.2.1 CIM_Processor.SystemCreationClassName**

2990 The name of the class or the subclass used in the creation of an instance of the containing
2991 system. This will typically be “CIM_ComputerSystem”.

2992 Formulate as a string:

2993 <”CIM_ComputerSystem” or the name of the subclass created>

2994

2995 **10.8.2.2 CIM_Processor.CreationClassName**

2996 Per the DMTF definition of the class property: The name of the class or the subclass used in the
2997 creation of an instance. This will typically be “CIM_Processor”.

2998 Formulate as a string:

2999 <”CIM_Processor” or the name of the subclass created>

3000 **10.8.2.3 CIM_Processor.SystemName**

3001 The host name of the containing system. Derived from the CIM_ComputerSystem.Name
3002 property of the instance representing the containing system.

3003 Formulate as a string:

3004 <CIM_ComputerSystem.Name>

3005 **10.8.2.4 CIM_Processor.DeviceID**

3006 Formulate as string:

3007 (string)<Entity ID from SDR> || “:” || (string)<Entity Instance from SDR> || “:” ||

3008 || (string)<Sensor Number from SDR> || “:” || <”Processor”>

3009 **10.8.2.5 CIM_Processor.UniqueID**

3010 **10.8.2.6 CIM_Processor.EnabledState**

3011 Formulate as string:

3012

3013 If Sensor offset 08h (Processor Disabled) = 1b

3014 <”Disabled”>

3015 Else

3016 <”Enabled”>

3017 10.8.2.7 CIM_Processor.Description

3018 Formulate as string:

3019 (string) “Processor” || “(” || (string)(SensorNumber) || “.” || (string)(OwnerLUN) || “.” ||
 3020 string)(OwnerID) || “)” || “ for ” || (string)(<IPMI_SDR_EntityID>) || “ “ ||
 3021 (string)(<IPMI_SDR_EntityInstanceNumber>)

3022 10.8.2.8 CIM_Processor.ElementName

3023 Formulate as string:

3024 (string)< SDR entry bytes 49+, see IPMI v1.5 Table 37-1> || “(” || (string)<Entity ID from SDR>
 3025 || “.” || (string)<Entity Instance from SDR> || “)”

3026

3027 10.8.2.9 CIM_Processor.HealthState**3028 10.8.2.10 CIM_Processor.OperationalStatus****3029 10.9 CIM_Fan Mapping****3030 10.9.1 CIM_Fan Methods**

3031 No Mapping Defined.

3032 10.9.2 CIM_Fan Properties

3033 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
 3034 properties that MAY be present. For example, there may be properties that are only filled out in
 3035 special configurations of the CIM Schema.

3036 Properties that are listed in the Required section in the following table MUST be implemented as
 3037 defined in the subsections following this table unless the Notes column for each property
 3038 specifies the defining reference.

3039

Table 23 CIM_Fan Class Properties

Required Properties	Notes
SystemCreationClassName	Key
SystemName	Key
CreationClassName	Key
DeviceID	Key
HealthState	
OperationalStatus	
Description	
ElementName	
<i>Recommended Properties</i>	Notes
VariableSpeed	
DesiredSpeed	
OtherIdentifyingInfo	
AdditionalAvailability	
IdentifyingDescriptions	
AdditionalAvailability	
EnabledState	
OtherEnabledState	
RequestedState	
EnabledDefault	
TimeOfLastStateChange	
Name	
StatusDescriptions	
InstallDate	
ActiveCooling	
Caption	

3040

3041 10.9.2.1 CIM_Fan.SystemCreationClassName

3042 The name of the class or the subclass used in the creation of an instance of the containing
 3043 system. This will typically be “CIM_ComputerSystem”.

3044 Formulate as a string:

3045 <”CIM_ComputerSystem” or the name of the subclass created>

3046

3047 10.9.2.2 CIM_Fan.CreationClassName

3048 Per the DMTF definition of the class property: The name of the class or the subclass used in the
 3049 creation of an instance. This will typically be “CIM_Fan”.

3050 Formulate as a string:

3051 <"CIM_Fan" or the name of the subclass created>

3052 **10.9.2.3 CIM_Fan.SystemName**

3053 The host name of the containing system. Derived from the CIM_ComputerSystem.Name
3054 property of the instance representing the containing system.

3055 Formulate as a string:

3056 <CIM_ComputerSystem.Name>

3057 **10.9.2.4 CIM_Fan.DeviceID**

3058 Formulate as string:

3059 (string)<Entity ID from SDR> || ":" || (string)<Entity Instance from SDR> || ":" ||

3060 || (string)<Sensor Number from SDR> || ":" || <"Fan">

3061

3062 **10.9.2.5 CIM_Fan.Description**

3063 Formulate as string:

3064 "Fan" || "(" || (string)(SensorNumber) || "." || (string)(OwnerLUN) || "." ||
3065 string)(OwnerID) || ")" || " for " || (string)(<IPMI_SDR_EntityID>) || "" ||
3066 (string)(<IPMI_SDR_EntityInstanceNumber>)

3067 **10.9.2.6 CIM_Fan.ElementName**

3068 Formulate as string:

3069 (string)< SDR entry bytes 49+, see IPMI v1.5 Table 37-1> || "(" || (string)<Entity ID from
3070 SDR> || ":" || (string)<Entity Instance from SDR> || ")"

3071 **10.9.2.7 CIM_Fan.HealthState**

3072 **10.9.2.8 CIM_Fan.OperationalStatus**

3073 **10.10 CIM_PowerSupply Mapping**

3074 **10.10.1 CIM_PowerSupply Methods**

3075 No Mapping Defined.

3076 **10.10.2 CIM_PowerSupply Properties**

3077 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
 3078 properties that MAY be present. For example, there may be properties that are only filled out in
 3079 special configurations of the CIM Schema.

3080 Properties are that listed in the Required section in the following table below MUST be
 3081 implemented as defined in the subsections following this table unless the Notes column for each
 3082 property specifies the defining reference.

3083 **Table 24 CIM_PowerSupply Class Properties**

Required Properties	Notes
SystemCreationClassName	Key
SystemName	Key
CreationClassName	Key
DeviceID	Key
HealthState	
OperationalStatus	
Description	
ElementName	
<i>Recommended Properties</i>	<i>Notes</i>

3084
 3085 **10.10.2.1 CIM_PowerSupply.SystemCreationClassName**

3086 The name of the class or the subclass used in the creation of an instance of the containing
 3087 system. This will typically be “CIM_ComputerSystem”.

3088 Formulate as a string:

3089 **10.10.2.2 <“CIM_ComputerSystem” or the name of the subclass created>**
 3090 **CIM_PowerSupply.CreationClassName**

3091 Per the DMTF definition of the class property: The name of the class or the subclass used in the
 3092 creation of an instance. This will typically be “CIM_PowerSupply”.

3093 Formulate as a string:

3094 <“CIM_PowerSupply” or the name of the subclass created>

3095 **10.10.2.3 CIM_PowerSupply.SystemName**

3096 The host name of the containing system. Derived from the CIM_ComputerSystem.Name
 3097 property of the instance representing the containing system.

3098 Formulate as a string:

- 3099 <CIM_ComputerSystem.Name>
- 3100 **10.10.2.4 CIM_PowerSupply.DeviceID**
- 3101 Formulate as string:
- 3102 (string)<Entity ID from SDR> || “.” || (string)<Entity Instance from SDR> || “.” ||
- 3103 || (string)<Sensor Number from SDR> || “.” || <”Power Supply”>
- 3104 **10.10.2.5 CIM_PowerSupply.Description**
- 3105 Formulate as string:
- 3106 “Power Supply” || “(” || (string)(SensorNumber) || “.” || (string)(OwnerLUN) || “.” ||
- 3107 string)(OwnerID) || “)” || “ for ” || (string)(<IPMI_SDR_EntityID>) || “ “ ||
- 3108 (string)(<IPMI_SDR_EntityInstanceNumber>)
- 3109 **10.10.2.6 CIM_PowerSupply.ElementName**
- 3110 Formulate as string:
- 3111 (string)< SDR entry bytes 49+, see IPMI v1.5 Table 37-1> || “(” || (string)<Entity ID
- 3112 from SDR> || “.” || (string)<Entity Instance from SDR> || “)”
- 3113 **10.10.2.7 CIM_PowerSupply.HealthState**
- 3114 **10.10.2.8 CIM_PowerSupply.OperationalStatus**
- 3115 **10.11 CIM_Memory Mapping**
- 3116 **10.11.1 CIM_Memory Methods**
- 3117 No Mapping Defined.
- 3118 **10.11.2 CIM_Memory Properties**
- 3119 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
- 3120 properties that MAY be present. For example, there may be properties that are only filled out in
- 3121 special configurations of the CIM Schema.
- 3122 Properties that are listed in the Required section in the following table MUST be implemented as
- 3123 defined in the subsections following this table unless the Notes column for each property
- 3124 specifies the defining reference.

3125

Table 25 CIM_Memory Class Properties

Required Properties	Notes
SystemCreationClassName	Key
SystemName	Key
CreationClassName	Key
DeviceID	Key
HealthState	
OperationalStatus	
Description	
ElementName	

3126

Recommended Properties	Notes
Blocksize	
NumberOfBlocks	
Name	
NameFormat	
NameNamespace	
IdentifyingDescriptions	
EnabledState	
RequestedState	
EnabledDefault	
HealthState	
OperationalStatus	
StartingAddress	
EndingAddress	
Purpose	
Volatile	
ErrorMethodology	
ConsumableBlocks	
IsBasedOnUnderlyingRedundancy	
ExtentStatus	
NoSinglePointOfFailure	
DataRedundancy	

Recommended Properties	Notes
PackageRedundancy	
DeltaReservation	
Primordial	
OtherNameNamespace	
OtherNameFormat	
OtherIdentifyingInfo	
OtherEnabledState	
TimeOfLastStateChange	
InstallDate	
StatusDescriptions	
AdditionalAvailability	
Caption	

3127 **10.11.2.1 CIM_Memory.SystemCreationClassName**

3128 The name of the class or the subclass used in the creation of an instance of the containing
3129 system. This will typically be “CIM_ComputerSystem”.

3130 Formulate as a string:

3131 <”CIM_ComputerSystem” or the name of the subclass created>

3132 **10.11.2.2 CIM_Memory.CreationClassName**

3133 Per the DMTF definition of the class property: The name of the class or the subclass used in the
3134 creation of an instance. This will typically be “CIM_Memory”.

3135 Formulate as a string:

3136 <”CIM_Memory” or the name of the subclass created>

3137 **10.11.2.3 CIM_Memory.SystemName**

3138 The host name of the containing system. Derived from the CIM_ComputerSystem.Name
3139 property of the instance representing the containing system.

3140 Formulate as a string:

3141 <CIM_ComputerSystem.Name>

3142 **10.11.2.4 CIM_Memory.DeviceID**

3143 Formulate as string:

3144 (string)<Entity ID from SDR> || “:” || (string)<Entity Instance from SDR> || “:” ||

3145 || (string)<Sensor Number from SDR> || “:” || <”Memory”>

3146 **10.11.2.5 CIM_Memory.Description**

3147 Formulate as string:

3148 “Memory” || “(” || (string)(SensorNumber) || “.” || (string)(OwnerLUN) || “.” ||
3149 string)(OwnerID) || “)” || “ for ” || (string)(<IPMI_SDR_EntityID>) || “ “ ||
3150 (string)(<IPMI_SDR_EntityInstanceNumber>)

3151 **10.11.2.6 CIM_Memory.ElementName**

3152 Formulate as string:

3153 (string)< SDR entry bytes 49+, see IPMI v1.5 Table 37-1> || “(” || (string)<Entity ID
3154 from SDR> || “:” || (string)<Entity Instance from SDR> || “)”
3155

3156 **10.11.2.7 CIM_Memory.HealthState**

3157 **10.11.2.8 CIM_Memory.OperationalStatus**

3158 **10.12 CIM_Battery Mapping**

3159 **10.12.1 CIM_Battery Methods**

3160 No Mapping Defined.

3161 **10.12.2 CIM_Battery Properties**

3162 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
3163 properties that MAY be present. For example, there may be properties that are only filled out in
3164 special configurations of the CIM Schema.

3165 Properties that are listed in the Required section in the following table MUST be implemented as
3166 defined in the subsections following this table unless the Notes column for each property
3167 specifies the defining reference.

3168

Table 26 CIM_Battery Class Properties

Required Properties	Notes
SystemCreationClassName	Key
SystemName	Key
CreationClassName	Key
DeviceID	Key
BatteryStatus	
HealthState	
OperationalStatus	
Description	
ElementName	

3169

<i>Recommended Properties</i>	Notes
EnabledState	
RequestedState	
EnabledDefault	
Name	
Description	
EstimatedRunTime	
EstimatedChargeRemaining	
RechargeCount	
MaxRechargeCount	
MaximumError	
RemainingCapacity	
TimeOnBattery	
Chemistry	
DesignCapacity	
FullChargeCapacity	
DesignVoltage	
SmartBatteryVersion	
TimeToFullCharge	
ExpectedLife	
MaxRechargeTime	
TimeOfLastChange	
StatusDescriptions	
InstallDate	
Caption	
ElementName	
IdentifyingDescriptions	
AdditionalAvailability	
OtherIdentifyingInfo	

<i>Recommended Properties</i>	<i>Notes</i>
OtherEnabledState	

3170 **10.12.2.1 CIM_Battery.SystemCreationClassName**

3171 The name of the class or the subclass used in the creation of an instance of the containing
3172 system. This will typically be “CIM_ComputerSystem”.

3173 Formulate as a string:

3174 <”CIM_ComputerSystem” or the name of the subclass created>

3175

3176 **10.12.2.2 CIM_Battery.CreationClassName**

3177 Per the DMTF definition of the class property: The name of the class or the subclass used in the
3178 creation of an instance. This will typically be “CIM_Battery”.

3179 Formulate as a string:

3180 <”CIM_Battery” or the name of the subclass created>

3181 **10.12.2.3 CIM_Battery.SystemName**

3182 The host name of the containing system. Derived from the CIM_ComputerSystem.Name
3183 property of the instance representing the containing system.

3184 Formulate as a string:

3185 <CIM_ComputerSystem.Name>

3186 **10.12.2.4 CIM_Battery.DeviceID**

3187 Formulate as string:

3188 (string)<Entity ID from SDR> || “:” || (string)<Entity Instance from SDR> || “:” ||

3189 || (string)<Sensor Number from SDR> || “:” || <”Battery”>

3190 **10.12.2.5 CIM_Battery.BatteryStatus**

3191 **10.12.2.6 CIM_Battery.Description**

3192 Formulate as string:

3193 “Battery” || “(” || (string)(SensorNumber) || “.” || (string)(OwnerLUN) || “.” ||

3194 string)(OwnerID) || “)” || “ for ” || (string)(<IPMI_SDR_EntityID>) || “ “ ||

3195 (string)(<IPMI_SDR_EntityInstanceNumber>)

3196

3197 **10.12.2.7 CIM_Battery.ElementName**

3198 Formulate as string:

3199 (string)< SDR entry bytes 49+, see IPMI v1.5 Table 37-1> || “(” || (string)<Entity ID
 3200 from SDR> || “:” || (string)<Entity Instance from SDR> || “)”
 3201

3202 **10.12.2.8 CIM_Battery.HealthState**3203 **10.12.2.9 CIM_Battery.OperationalStatus**3204 **10.13 CIM_DiskDrive Mapping**3205 **10.13.1 CIM_DiskDrive Methods**

3206 No Mapping Defined.

3207 **10.13.2 CIM_DiskDrive Properties**

3208 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
 3209 properties that MAY be present. For example, there may be properties that are only filled out in
 3210 special configurations of the CIM Schema.

3211 Properties that are listed in the Required section in the table below MUST be implemented as
 3212 defined in the subsections following this table unless the Notes column for each property
 3213 specifies the defining reference.

3214 **Table 27 CIM_DiskDrive Class Properties**

Required Properties	Notes
SystemCreationClassName	Key
SystemName	Key
CreationClassName	Key
DeviceID	Key
HealthState	
OperationalStatus	
Description	
ElementName	
<i>Recommended Properties</i>	Notes
Capabilities	
CapabilityDescriptions	
ErrorMethodology	

3215

Recommended Properties	Notes
CompressionMethod	
NumberOfMediaSupported	
MaxMediaSize	
DefaultBlockSize	
MaxBlockSize	
MinBlockSize	
NeedsCleaning	
MediaIsLocked	
Security	
LastCleaned	
MaxAccessTime	
UncompressedDataRate	
LoadTime	
UnloadTime	
MountCount	
TimeOfLastMount	
TotalMountTime	
UnitsDescription	
MaxUnitsBeforeCleaning	
UnitsUsed	
EnabledState	
RequestedState	
EnabledDefault	
TimeOfLastStateChange	
Name	
Description	
InstallDate	
Caption	
IdentifyingDescriptions	
AdditionalAvailability	
OtherIdentifyingInfo	
OtherEnabledState	

3216 **10.13.2.1 CIM_DiskDrive.SystemCreationClassName**

3217 The name of the class or the subclass used in the creation of an instance of the containing
 3218 system. This will typically be “CIM_ComputerSystem”.

3219 Formulate as a string:

3220 <<”CIM_ComputerSystem” or the name of the subclass created>

- 3221
- 3222 **10.13.2.2 CIM_DiskDrive.CreationClassName**
- 3223 Per the DMTF definition of the class property: The name of the class or the subclass used in the
3224 creation of an instance. This will typically be “CIM_DiskDrive”.
- 3225 Formulate as a string:
- 3226 <”CIM_DiskDrive” or the name of the subclass created>
- 3227 **10.13.2.3 CIM_DiskDrive.SystemName**
- 3228 The host name of the containing system. Derived from the CIM_ComputerSystem.Name
3229 property of the instance representing the containing system.
- 3230 Formulate as a string:
- 3231 <CIM_ComputerSystem.Name>
- 3232 **10.13.2.4 CIM_DiskDrive.DeviceID**
- 3233 Formulate as string:
- 3234 (string)<Entity ID from SDR> || “:” || (string)<Entity Instance from SDR> || “:” ||
3235 || (string)<Sensor Number from SDR> || “:” || <”DiskDrive”>
- 3236 **10.13.2.5 CIM_DiskDrive.Description**
- 3237 Formulate as string:
- 3238 “DiskDrive” || “(” || (string)(SensorNumber) || “.” || (string)(OwnerLUN) || “.” ||
3239 string)(OwnerID) || “)” || “ for ” || (string)(<IPMI_SDR_EntityID>) || “ “ ||
3240 (string)(<IPMI_SDR_EntityInstanceNumber>)
- 3241 **10.13.2.6 CIM_DiskDrive.ElementName**
- 3242 Formulate as string:
- 3243 (string)< SDR entry bytes 49+, see IPMI v1.5 Table 37-1> || “(” || (string)<Entity ID from SDR>
3244 || “:” || (string)<Entity Instance from SDR> || “)”
- 3245 **10.13.2.7 CIM_DiskDrive.HealthState**
- 3246 **10.13.2.8 CIM_DiskDrive.OperationalStatus**
- 3247

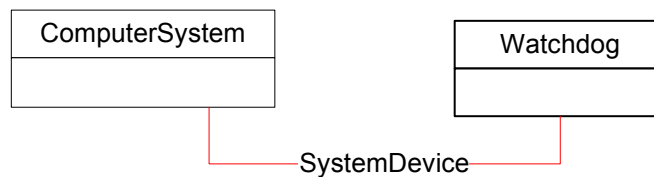
3248 11 IPMI Watchdog Mapping

3249 11.1 Introduction

3250 The IPMI Watchdog Mapping extends the management capability of the IPMI to CIM mapping
3251 by adding the capability to manage the IPMI Watchdog

3252 The following class diagram describes the CIM classes involved in representing the IPMI
3253 watchdog.

IPMI Watchdog Class Diagram



3254

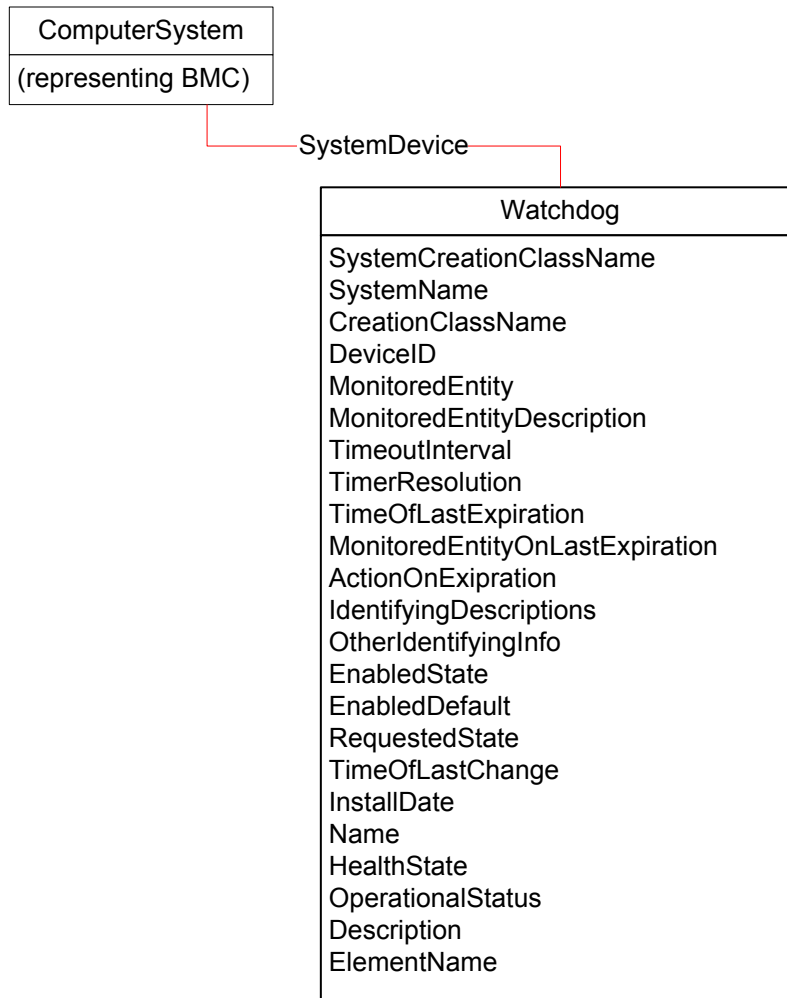
3255

Figure 18 Class Diagram – IPMI Watchdog

3256 11.2 Instance Diagrams

3257 The following instance diagram describes an instance of CIM_Watchdog associated with the
3258 instance of CIM_ComputerSystem representing the BMC.

IPMI Watchdog Instance Diagram



3259

3260

Figure 19 Instance Diagram – IPMI Watchdog

3261 11.3 Mapping Requirements

3262 11.4 CIM_Watchdog Mapping

3263 11.4.1 CIM_Watchdog Methods

3264 11.4.1.1 Method: Reset()

3265 `uint32 Reset()`

3266

3267 Per the DMTF definition: The return value should be 0 if the request was successfully executed,
3268 1 if the request is not supported and some other value if an error occurred. This method should
3269 map to the IPMI Reset Watchdog Timer command.

3270

3271 A return value of **4 – Failed** would indicate a Reset Watchdog Timer Command response data
3272 completion code of 80h meaning “Attempt to start an uninitialized watchdog” as indicated in
3273 IPMI v1.5 specification table 21-2.

3274 Return value:

3275 0 – Completed with no error

3276 1 – Not Supported

3277 2 – Unspecified Error

3278 3 – Timeout

3279 4 – Failed

3280 5 – Invalid Parameter

3281 6..0xFFFF – DMTF_Reserved

3282 0x1000..0x7FFF – Method_Reserved

3283 0x8000.. – Vendor_Reserved

3284

3285 **11.4.1.2 Method: RequestStateChange**

```
3286 uint32 RequestStateChange (  
3287     [IN] uint16 RequestedState,  
3288     [IN] datetime TimeoutPeriod)
```

3289

3290 TimeoutPeriod maps to the Initial Countdown Value bytes for the Set Watchdog Timer
3291 command. RequestedState is not used; state changes can be performed by writing appropriate
3292 properties in the CIM_Watchdog class instance.

3293 Return value:

3294 0 – Completed with no error

3295 1 – Not Supported

3296 2 – Unspecified Error

3297 3 – Timeout

3298 4 – Failed

3299 5 – Invalid Parameter

3300 6..0xFFFF – DMTF_Reserved

3301 0x1000..0x7FFF – Method_Reserved

3302 0x8000.. – Vendor_Reserved

3303

3304 **11.4.1.3 Method: KeepAlive**

```
3305 uint32 KeepAlive()
```

3306

3307 A method to re-arm the timer. This method is only used if the MonitoredEntity is "Application".
3308 It returns 0 if successful, 1 if unsupported, and any other value if an error occurred. In an IPMI
3309 CIM mapping this will always return **1 – Not Supported**.

3310 Return value:

3311 0 – Completed with no error

3312 1 – Not Supported

3313 **11.4.2 CIM_Watchdog Properties**

3314 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
3315 properties that MAY be present. For example, there may be properties that are only filled out in
3316 special configurations of the CIM Schema.

3317 Properties that are listed in the Required section in Table 6 below MUST be implemented as
3318 defined in the subsections following this table unless the Notes column for each property
3319 specifies the defining reference.

3320

Table 28 CIM_Watchdog Class Properties

Required Properties	Notes
SystemCreationClassName	Key
SystemName	Key
CreationClassName	Key
DeviceID	Key
MonitoredEntity	
MonitoredEntityDescription	
TimeoutInterval	
TimerResolution	
TimeOfLastExpiration	
MonitoredEntityOnLastExpiration	
ActionOnExpiration	
EnabledState	
DefaultState	
RequestedState	
Name	
OperationsStatus	
HealthState	
LogState	Pending CR Approval in the DMTF
ActionOnPretimeoutInterval	Pending CR Approval in the DMTF
PretimeoutInterval	Pending CR Approval in the DMTF

3321

<i>Recommended Properties</i>	Notes
OtherIdentifyingInfo	
IdentifyingDescriptions	
AdditionalAvailability	
OtherEnabledState	
TimeOfLastStateChange	
InstallDate	
Caption	
Description	
ElementName	

3322

3323 The following subsections describe procedures for generating values for various properties of
 3324 CIM classes required in this profile.

3325 **11.4.2.1 CIM_Watchdog.SystemCreationClassName**

3326 The name of the class or the subclass used in the creation of an instance of the containing
 3327 system. This will typically be “CIM_ComputerSystem”.

3328 Formulate as a string:

3329 <"CIM_ComputerSystem" or the name of the subclass created>

3330 **11.4.2.2 CIM_Watchdog.SystemName**

3331 The host name of the containing system. Derived from the CIM_ComputerSystem.Name
3332 property of the instance representing the containing system.

3333 Formulate as a string:

3334 <CIM_ComputerSystem.Name>

3335 **11.4.2.3 CIM_Watchdog.CreationClassName**

3336 Per the DMTF definition of the class property: The name of the class or the subclass used in the
3337 creation of an instance. This will typically be "CIM_Watchdog".

3338 Formulate as a string:

3339 <"CIM_WatchDog" or the name of the subclass created>

3340 **11.4.2.4 CIM_Watchdog.DeviceID**

3341 Formulate as a string:

3342 An address or other identifying information to uniquely name the LogicalDevice.

3343 **11.4.2.5 CIM_Watchdog.MonitoredEntity**

3344

3345 Values { "Unknown" , "Other" , "Operating System" , "Operating System Boot Process" ,
3346 "Operating System Shutdown Process" , "Firmware Boot Process" , "BIOS Boot Process" ,
3347 "Application" , "Service Processor" }

3348 ValueMap { "0" , "1" , "2" , "3" , "4" , "5" , "6" , "7" , "8" }

3349 The entity that is currently being monitored by the WatchDog. This property is used to identify
3350 the module that is responsible for or whose information is used to re-arm the watchdog at
3351 periodic intervals.

3352 IPMI supported entities map to MonitoredEntity values as follows:

001b BIOS FRB2	"6" "BIOS Boot Process"
010b BIOS POST	"5" "Firmware Boot Process"
011b OS Load	"3" "Operating System Boot Process"
100b SMS/OS	"2" "Operating System"
101b OEM	"1" "Other"

3353

3354 MonitoredEntity values not listed above are not supported in this mapping.

3355 **11.4.2.6 CIM_Watchdog.MonitoredEntityDecription**

3356 Formulate as string:

3357 A string describing more textual information about the monitored entity:

3358 “BIOS FRB2”, “BIOS POST”, “OS Load”, “SMS/OS”, “OEM”

3359 **11.4.2.7 CIM_Watchdog.TimeoutInterval**

3360

3361 Units ("MicroSeconds")

3362

3363 The timeout interval used by the watchdog, in MicroSeconds.

3364 **11.4.2.8 CIM_Watchdog.TimerResolution**

3365

3366 Units ("MicroSeconds")

3367

3368 Resolution of the timer. For example, if this value is 100, then the timer can expire anytime
3369 between (TimeoutInterval- 100) microseconds or (TimeoutInterval+100) microseconds.

3370

3371 IPMI v1.5 Watchdog resolution is one count/100 milliseconds, or 100000 microseconds. The
3372 TimerResolution value will always map to 100000.

3373 **11.4.2.9 CIM_Watchdog.TimeOfLastExpiration**

3374 Datetime

3375

3376 The time of the last timer expiry.

3377 **11.4.2.10 CIM_Watchdog.MonitoredEntityOnLastExpiration**

3378

3379 Values { "Unknown" , "Other" , "Operating System" , "Operating System Boot Process" ,
3380 "Operating System Shutdown Process" , "Firmware Boot Process" , "BIOS Boot Process" ,
3381 "Application" , "Service Processor" }

3382

3383 ValueMap { "0" , "1" , "2" , "3" , "4" , "5" , "6" , "7" , "8" }

3384

3385 Monitored entity at the time of last timer expiry.

3386 IPMI supported entities map to MonitoredEntityOnLastExpiration values as follows:

001b BIOS FRB2	“6” “BIOS Boot Process”
010b BIOS POST	“5” “Firmware Boot Process”

011b OS Load	“3” “Operating System Boot Process”
100b SMS/OS	“2” “Operating System”
101b OEM	“1” “Other”

3387

3388 MonitoredEntityOnLastExpiration values not listed above are not supported in this mapping.

3389 **11.4.2.11 CIM_Watchdog.ActionOnExpiration**

3390

3391 Values { "None - Status Only" , "System Reset" , "System Power Off" , "System Power Off, then
3392 On" , "Generate System NonMaskableInterrupt (NMI)" , "Generate System Management
3393 Interrupt (SMI)" }

3394

3395 ValueMap { "0" , "1" , "2" , "3" , "4" , "5" }

3396

3397 The action that should happen upon the expiry of the watchdog. IPMI supports timeout actions
3398 as defined for bits [2:0] of the Set Watchdog Timer/Get Watchdog Timer commands' Timer
3399 Action byte. These map directly to ActionOnExpiration values as follows:

3400

000b No Action	“0” “None – Status Only”
001b Hard Reset	“1” “ System Reset”
010b Power Down	“2” “System Power Off”
011b Power Cycle	“3” “System Power Off, then On”

3401

3402 Values “4” and “5” are not supported in this mapping.

3403 **11.4.2.12 CIM_Watchdog.EnabledState**

3404

3405 Values { "Unknown" , "Other" , "Enabled" , "Disabled" , "Shutting Down" , "Not Applicable" ,
3406 "Enabled but Offline" , "In Test" , "Deferred" , "Quiesce" , "Starting" , "DMTF Reserved" ,
3407 "Vendor Reserved" }

3408

3409 ModelCorrespondence { "CIM_EnabledLogicalElement.OtherEnabledState" }

3410

3411 ValueMap { "0" , "1" , "2" , "3" , "4" , "5" , "6" , "7" , "8" , "9" , "10" , "11..32767" ,
3412 "32768..65535" }

3413

3414 **EnabledState** = 5 - Not Applicable (5) indicates the element doesn't support being
3415 enabled/disabled.

3416

3417 **11.4.2.13 CIM_Watchdog.DefaultState**

3418 Formulate as string:

3419 “”

3420 **11.4.2.14 CIM_Watchdog.RequestedState**

3421
3422 Values { "Enabled" , "Disabled" , "Shut Down" , "No Change" , "Offline" , "Test" , "Deferred" ,
3423 "Quiesce" , "Reboot" , "Reset" , "Not Applicable" , "DMTF Reserved" , "Vendor Reserved" }

3424
3425 ModelCorrespondence { "CIM_EnabledLogicalElement.EnabledState" }

3426
3427 ValueMap { "2" , "3" , "4" , "5" , "6" , "7" , "8" , "9" , "10" , "11" , "12" , ".." , "32768..65535" }

3428
3429 **RequestedState** = 12 - Not supported

3430
3431

3432 **11.4.2.15 CIM_Watchdog.Name**

3433 Per the DMTF definition, the Name property defines the label by which the object is known.
3434 When subclassed, the Name property can be overridden to be a Key property.

3435 Formulate as string:

3436 "IPMI_WatchDog"

3437 **11.4.2.16 CIM_Watchdog.OperationalStatus**

3438
3439 Values { "Unknown" , "Other" , "OK" , "Degraded" , "Stressed" , "Predictive Failure" , "Error" ,
3440 "Non-Recoverable Error" , "Starting" , "Stopping" , "Stopped" , "In Service" , "No Contact" ,
3441 "Lost Communication" , "Aborted" , "Dormant" , "Supporting Entity in Error" , "Completed" ,
3442 "Power Mode" , "DMTF Reserved" , "Vendor Reserved" }

3443 ModelCorrespondence { "CIM_ManagedSystemElement.StatusDescriptions" }

3444 ValueMap { "0" , "1" , "2" , "3" , "4" , "5" , "6" , "7" , "8" , "9" , "10" , "11" , "12" , "13" , "14" ,
3445 "15" , "16" , "17" , "18" , ".." , "0x8000.." }

3446 ArrayType ("Indexed")

3447
3448 Only the first element of OperationalStatus array is used in this mapping. OperationalStatus
3449 Values map to return states from the IPMI v1.5 Get Watchdog Timer command as follows:

3450
3451

Timer Use bit [6] = 1b "timer is started (running)"	"2", "OK"
Timer Use bit [6] = 0b "timer is stopped"	"10", "Stopped"
Error completion code from Get Watchdog Timer command	"6", "Error"
No response from any Get Watchdog Timer command	"12", "No Contact"
No response from most recent Get Watchdog Timer command	"13", "Lost Communication"

3451

3452 **11.4.2.17 CIM_Watchdog.HealthState**

3453
3454 Values { "Unknown" , "OK" , "Degraded/Warning" , "Minor failure" , "Major failure" , "Critical

3455 failure" , "Non-recoverable error" , "DMTF Reserved" }
 3456 ValueMap { "0" , "5" , "10" , "15" , "20" , "25" , "30" , ".." }
 3457
 3458 “0” – Not supported
 3459

3460 **11.4.2.18 CIM_Watchdog.LogState**

3461
 3462 Values { "Unknown" , "Log" , "Do Not Log" }
 3463 ValueMap { "0" , "1" , "2" }
 3464
 3465 Maps to Timer Use bit [7] indicating logging status for the watchdog
 3466

3467 **11.4.2.19 CIM_Watchdog.ActionOnPretimeoutInterval**

3468
 3469 Values { "None - Status Only" , "Generate System Management Interrupt (SMI)" , "Generate
 3470 System NonMaskableInterrupt (NMI)" , "Messaging Interrupt" }
 3471
 3472 ValueMap { "0" , "1" , "2" , "3" }
 3473
 3474 The action that should happen when the pretimeout interval for the watchdog timer is passed.
 3475 IPMI supports timeout actions as defined for bits [2:0] of the Set Watchdog Timer/Get Watchdog
 3476 Timer commands' Timer Action byte. These map to ActionOnPretimeoutInterval values as
 3477 follows:
 3478

000b No Action	“0” “None – Status Only”
001b SMI	“1” , "Generate System Management Interrupt (SMI)"
010b NMI/Diagnostic Interrupt	“2” “Generate System NonMaskableInterrupt (NMI)”
011b Messaging Interrupt	“3” “Messaging Interrupt”

3479
 3480

3481 **11.4.2.20 CIM_Watchdog.PretimeoutInterval**

3482
 3483 Units ("Seconds")
 3484
 3485 The pretimeout interval used by the IPMI watchdog, in Seconds.
 3486

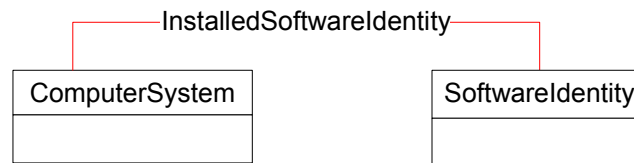
3487 12 IPMI Software Identity Mapping

3488 12.1 Introduction

3489 The IPMI Software Identity Mapping extends the management capability of the IPMI to CIM
3490 mapping by adding the capability to represent software and firmware known to the BMC
3491 (including the BMC firmware) where information about the firmware is exposed by the BMC.

3492 The following class diagram describes the CIM classes involved in representing the software and
3493 firmware.

IPMI Software Identity Mapping Class Diagram



3494

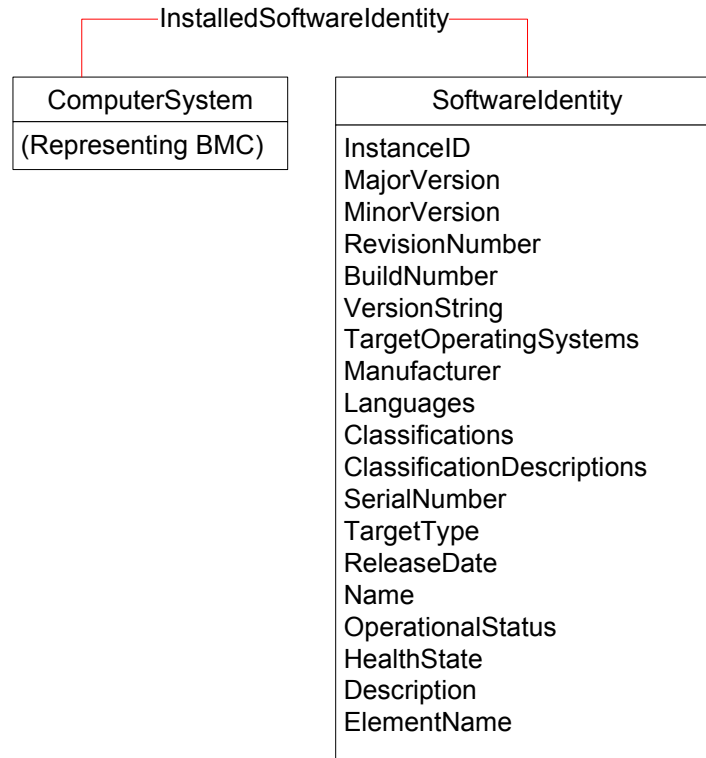
3495

Figure 20 Class Diagram – Software Identity

3496 12.2 Instance Diagrams

3497 The following instance diagram describes an instance of CIM_SoftwareIdentity associated with
3498 the instance of CIM_ComputerSystem representing the BMC.

IPMI Software Identity Mapping Instance Diagram (BMC Firmware)



3499

3500

Figure 21 Instance Diagram – Software Identity

3501 12.3 Mapping Requirements

3502 12.4 CIM_SoftwareIdentity Mapping

3503 12.4.1 CIM_SoftwareIdentity Methods

3504 None Defined.

3505 12.4.2 CIM_SoftwareIdentity Properties

3506 REQUIRED properties are properties that MUST be present. RECOMMENDED properties are
 3507 properties that MAY be present. For example, there may be properties that are only filled out in
 3508 special configurations of the CIM Schema.

3509 Properties that are listed in the Required section in Table 6 below MUST be implemented as
 3510 defined in the subsections following this table unless the Notes column for each property
 3511 specifies the defining reference.

3512

Table 29 CIM_SoftwareIdentity Class Properties

Required Properties	Notes
InstanceID	Key
RevisionNumber	
VersionString	
Manufacturer	
Name	
OperationalStatus	
HealthState	
<i>Recommended Properties</i>	Notes
TargetOperatingSystems	
ReleaseDate	
BuildNumber	
MajorVersion	
MinorVersion	
Languages	
Classifications	
ClassificationDescriptions	
SerialNumber	
TargetType	
Caption	
Description	
ElementName	

3513

3514

3515 The following subsections describe procedures for generating values for various properties of
 3516 CIM classes required in this profile.

3517 **12.4.2.1 CIM_SoftwareIdentity.InstanceID**

3518 Per the DMTF definition of the class property: In order to ensure uniqueness within the
 3519 NameSpace, the value of InstanceID SHOULD be constructed using the following ' preferred '
 3520 algorithm: < OrgID > : < LocalID > Where < OrgID > and < LocalID > are separated by a colon
 3521 ' : ' , and where < OrgID > MUST include a copyrighted, trademarked or otherwise unique name
 3522 that is owned by the business entity creating/defining the InstanceID, or is a registered ID that is
 3523 assigned to the business entity by a recognized global authority.

3524 Formulate as a string:

3525 <"IPMI:" || DeviceID from the Get Device ID Command>

3526

3527 12.4.2.2 CIM_SoftwareIdentity.RevisionNumber

3528 Per the DMTF definition of the class property: This property is defined as a numeric value to
3529 allow the determination of 'newer' vs. 'older' releases. A 'newer' revision is indicated by a
3530 larger numeric value.

3531 Formulate as a string:

3532 <Firmware Revision 1 from the Get Device ID Command || “. ”

3533 || Firmware Revision 2 from the Get Device ID Command>

3534 12.4.2.3 CIM_SoftwareIdentity.VersionString

3535 Per the DMTF definition of the class property: A string representing the complete software
3536 version information.

3537 Formulate as a string:

3538 <<Firmware Revision 1 from the Get Device ID Command || “. ”

3539 || Firmware Revision 2 from the Get Device ID Command || “. ”

3540 || Auxillary Firmware Revision Information from the Get Device ID Command>

3541 12.4.2.4 CIM_SoftwareIdentity.Manufacturer

3542 Per the DMTF definition of the class property: Manufacturer of this software.

3543 Formulate as a string:

3544 <<Manufacturer ID from the Get Device ID Command>

3545 12.4.2.5 CIM_SoftwareIdentity.Name

3546 Formulate as a string:

3547 < “IPMI BMC Firmware “ ||

3548 || Firmware Revision 1 from the Get Device ID Command || “. ”

3549 || Firmware Revision 2 from the Get Device ID Command>

3550 12.4.2.6 CIM_SoftwareIdentity.OperationalStatus**3551 12.4.2.7 CIM_SoftwareIdentity.HealthState**

3552

3553

3554 **13 Acknowledgments**

3555 **Editor**

3556 Jon Hass - Dell Inc. for the IPMI CIM Subcommittee of the IPMI Forum

3557

3558 **Contributors**

3559 Jon Hass - Dell Inc.

3560 Steffen C. Hulegaard - OSA Inc.

3561 Frank Li – OSA Inc.

3562 Arvind Kumar – Intel Corporation

3563 Tom Slaight - Intel Corporation

3564 Wayne Weilnau – Dell Inc.

3565 Chandra Mugunda – Dell Inc.

3566 Phil Chidester - Dell Inc.

3567 Steve Lyle – Hewlett-Packard Company

3568 Kevin Howard – Hewlett-Packard Company

3569

3570 **End of Document**