Document Number: DSP1033

Date: 2014-05-22

Version: 1.1.0

Profile Registration Profile

Document Type: Specification

Document Status: DMTF Standard

Document Language: en-US

Copyright notice

11 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, 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.

12 Implementation of certain elements of this standard or proposed standard 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patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a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.

13 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.

14

CONTENTS

15

Figures

16

Tables

17

Foreword

This document was prepared by the DMTF Architecture Working Group

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

Acknowledgements

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

- Andreas Maier, IBM (editor of this version)
- Jim Davis, WBEM Solutions
- George Ericson, EMC
- Steve Hand, Symantec
- Jon Hass, Dell Inc. (editor of prior versions)
- John Leung, Intel
- Aaron Merkin, IBM
- Khachatur Papanyan, Dell
- Karl Schopmeyer, Inova
- Christina Shaw, Hewlett-Packard Company
- Paul von Behren, Symantec
- Mike Walker, IBM

Introduction

34 This document defines the CIM model for discovering implemented profiles in a managed environment. The information in this document is intended to be sufficient for a provider or consumer of this data to identify unambiguously the classes, properties, methods, and values that need to be instantiated and manipulated.

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

36

Document conventions

37

Typographical conventions

38 The following typographical conventions are used in this document:

39 • Document titles are marked in *italics*.

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

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

42

OCL usage conventions

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

44 OCL statements are in `monospaced font`.

45

Deprecated material

46 Deprecated material is not recommended for use in new development efforts. Existing and new implementations may use this material, but they shall move to the favored approach as soon as possible. CIM services shall implement any deprecated elements as required by this document in order to achieve backwards compatibility. Although CIM clients may use deprecated elements, they are directed to use the favored elements instead.

47 Deprecated material should contain references to the last published version that included the deprecated material as normative material and to a description of the favored approach.

48 The following typographical convention indicates deprecated material:

49

DEPRECATED

50 Deprecated material appears here.

51 **DEPRECATED**

---

52 In places where this typographical convention cannot be used (for example, tables or figures), the "DEPRECATED" label is used alone.

53

Profile Registration Profile

54

1 Scope

55 The Profile Registration profile extends the management capabilities of referencing profiles by adding the capabilities to advertise conformance of the implementation to the referencing profiles, and to discover instances for which conformance to the referencing profile is advertised.

56

2 Normative references

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

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

59 DMTF DSP0223, *Generic Operations 1.0*, http://www.dmtf.org/standards/published_documents/DSP0223_1.0.pdf

60 DMTF DSP1001, *Management Profile Specification Usage Guide 1.0*, http://www.dmtf.org/standards/published_documents/DSP1001_1.0.pdf

61 DMTF DSP1023, *Software Inventory Profile 1.0*, http://www.dmtf.org/standards/published_documents/DSP1023_1.0.pdf

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

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

64

3 Terms and definitions

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

66

3.1 General

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

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

69

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

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

70 The following additional terms are defined in this document.

71 **3.2**

72 **autonomous profile**

73 A profile that addresses an autonomous and self-contained management domain. For a complete definition, see DSP1001.

74 DSP1001 defines that in autonomous profiles, the central class adaptation and scoping class adaptation are the same. Thus, autonomous profiles cannot be scoped by other profiles. With the exception of this profile, autonomous profiles do not need to be referenced in order to be implemented, and can therefore be implemented alone. Autonomous profiles may reference component profiles and autonomous profiles (including themselves) and may scope component profiles. See also term "component profile".

75 **3.3**

76 **central class adaptation**

A class adaptation whose instances act as an algorithmic focal point for advertising conformance of an implementation to a profile. For a more general definition, see DSP1001. See also term "scoping class adaptation".

77 **3.4**

78 **central class methodology**

An algorithm for advertising profile conformance that uses the central instances of the registered profile as an algorithmic focal point. For a complete definition, see 6.3.2. See also term "scoping class methodology".

79 **3.5**

80 **central element**

The managed object type modeled by a central class adaptation. See also term "scoping element".

81 **3.6**

82 **central instance**

An instance of the central class adaptation. See also term "scoping instance".

83 **3.7**

84 **component profile**

85 A profile that addresses a subset of a management domain. For a complete definition, see DSP1001.

86 DSP1001 defines that in component profiles, the central class adaptation and scoping class adaptation are not the same. Component profiles need to be scoped by one or more scoping profiles to be implemented, and can be implemented only together with one of their scoping profiles. Component profiles may reference autonomous profiles and component profiles (including themselves) and may scope other component profiles. See also term "autonomous profile".

87 **3.8**

88 **Interop namespace**

89 A role of a CIM namespace for the purpose of providing a common and well-known place for clients to discover modeled entities, such as the profiles to which an implementation advertises conformance. The

term is also used for namespaces that assume that role. For a complete definition, see 6.4.1. See also term "implementation namespace".

90 **3.9**

91 **implementation namespace**

92 A role of a CIM namespace for the purpose of providing a place for CIM objects for which no specific namespace requirements are defined. The term is also used for namespaces that assume that role. For a complete definition, see 6.4.2. See also term "Interop namespace".

93 **3.10**

94 **profile**

A management profile, as defined in DSP1001.

95 **3.11**

96 **profile conformance**

97 Conformance of an implementation to one or more profiles, such that the implementation satisfies the rules for *full implementation conformance* defined in subclause 5.2.2 of DSP1001.

98 **3.12**

99 **referenced profile**

A profile that is listed in the profile references table of another or the same profile. For a complete definition, see subclause 7.9.1 of DSP1001.

100 **3.13**

101 **referencing profile**

A profile that lists the same or another profile in its profile references table. For a complete definition, see subclause 7.9.1 of DSP1001.

102 **3.14**

103 **registered profile**

104 A profile to which an implementation advertises conformance. Before version 1.1 of this profile, registered profiles were termed "subject profiles" (that term is now deprecated).

105 **3.15**

106 **scoping class adaptation**

A class adaptation that acts as an algorithmic focal point for advertising conformance of an implementation to a profile when using the scoping class methodology. For a more general definition, see DSP1001. See also term "central class adaptation".

107 **3.16**

108 **scoping class methodology**

An algorithm for advertising profile conformance that uses the scoping instances of the registered profile as an algorithmic focal point. For a complete definition, see 6.3.3. See also term "central class methodology".

109 **3.17**

110 **scoping element**

The managed object type modeled by a scoping class adaptation. See also term "central element".

111

3.18

112 **scoping instance**

An instance of the scoping class adaptation. See also term "central instance".

113 **3.19**

114 **scoping path**

An association traversal path between the central class adaptation and the scoping class adaptation. For a complete definition, see DSP1001.

115 **3.20**

116 **scoping profile**

117 A profile that provides a scope to a scoped profile by defining a central class adaptation that is based on the scoping class adaptation defined in the scoped profile. For a complete definition, see DSP1001.

118 **3.21**

119 **subject profile**

120 **DEPRECATED:** The term "subject profile" has been deprecated in version 1.1 of this profile, because its meaning as defined in this profile was different from the meaning as defined in DSP1001.

121 Use the term "registered profile" instead.

122

4 Symbols and abbreviated terms

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

124 This document does not define any additional abbreviations.

125

5 Synopsis

126 **Profile name:** Profile Registration

127 **Version:** 1.1.0

128 **Organization:** DMTF

129 **Abstract:** No

130 **Profile type:** Autonomous

131 **Schema:** DMTF CIM 2.39

132 **Central class adaptation:** RegisteredProfile

133 **Scoping class adaptation:** RegisteredProfile

134 The Profile Registration profile extends the management capabilities of referencing profiles by adding the capabilities to advertise and discover conformance of the implementation to the referencing profiles.

135 For historical reasons, the scoping and central class adaptations of the Profile Registration profile are the same, which qualifies it as an autonomous profile (version 1.0 of this profile was silent about that). Contrary to the usual ability of an autonomous profile to be implementable on its own, this profile can be implemented only in context of its referencing profile(s).

136 Table 1 identifies the profile references defined in this profile.

137

Table 1 – Profile references

Profile reference name	Profile name	Organi-zation	Version	Relation-ship	Description
SelfPRP	Profile Registration	DMTF	1.1	Mandatory	Used to advertise conformance of the implementation to this profile.
RefPRP	Profile Registration	DMTF	1.1	Mandatory	Used to advertise conformance of the implementation to a profile referenced by the registered profile.

138 Table 2 identifies the features defined in this profile.

139 **Table 2 – Features**

Feature	Requirement	Description
CentralClassMethodology	Optional	See 7.1.1.
GetCentralInstancesMethodology	Optional	See 7.1.2.
SoftwareIdentity	Optional	See 7.1.3.

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

141 **Table 3 – Adaptations**

Adaptation	Elements	Requirement	Description
Instantiated, embedded and abstract adaptations			
RegisteredProfile	CIM_RegisteredProfile	Mandatory	See 7.2.2.
ElementConformsToProfile	CIM_ElementConformsToProfile	ConditionalExclusive	See 7.2.3.
ScopingElement	CIM_ManagedElement	See derived adaptations	See 7.2.4.
CentralElement	CIM_ManagedElement	See derived adaptations	See 7.2.5.
ReferencedProfile	CIM_ReferencedProfile	Mandatory	See 7.2.6.
ReferencedRegisteredProfile	CIM_RegisteredProfile	Mandatory	See 7.2.7.
SoftwareIdentity	CIM_SoftwareIdentity	Conditional	See 7.2.8.
ElementSoftwareIdentity	CIM_ElementSoftwareIdentity	Conditional	See 7.2.9.
Indications and exceptions			
This profile does not define any such adaptations.			

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

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

Name	Description
State description: SimpleStateDescription	See 8.1.
Use case: RetrieveProfileInformationForComputerSystem	See 8.2.
Use case: RetrieveProfileVersionForFan	See 8.3.
Use case: RetrieveProfileVersionForPowerSupply	See 8.4.
Use case: AlgorithmForRetrievingProfileInformation	See 8.5.
Use case: DetermineConformingInstances	See 8.6.
Use case: AlgorithmForDeterminingAdvertisedProfiles	See 8.7.

Name	Description
Use case: AlgorithmForDeterminingTopLevelProfiles	See 8.8.
Use case: DetermineCentralInstancesForFan	See 8.9.
Use case: DetermineCentralInstancesForPowerSupply	See 8.10.
Use case: AlgorithmForDeterminingCentralInstancesOfProfile	See 8.11.
Use case: AlgorithmForDeterminingCentral	See 8.12.
State description: PeerComponentProfileStateDescription	See 8.13.
State description: ProfileComplianceHierarchyStateDescription	See 8.14.
State description: ProfileDerivationStateDescription	See 8.15.

6 Description

6.1 Profile relationships

The example in Figure 1 shows two important relationships between profiles that are used throughout this profile (the Profile Registration profile):

- The Fan profile (and similarly, the Sensors profile) is a *registered profile* from the perspective of its Profile Registration profile; that is, it is the profile that is advertised through its Profile Registration profile.
- The Sensors profile is a *referenced profile* from the perspective of the Fan profile; that is, it is listed in the profile references table of the Fan profile.

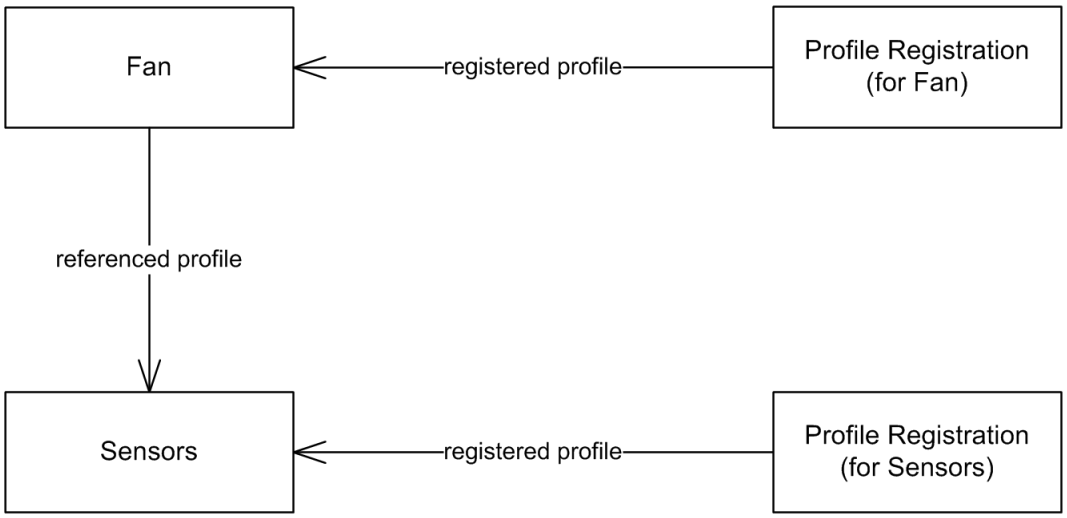
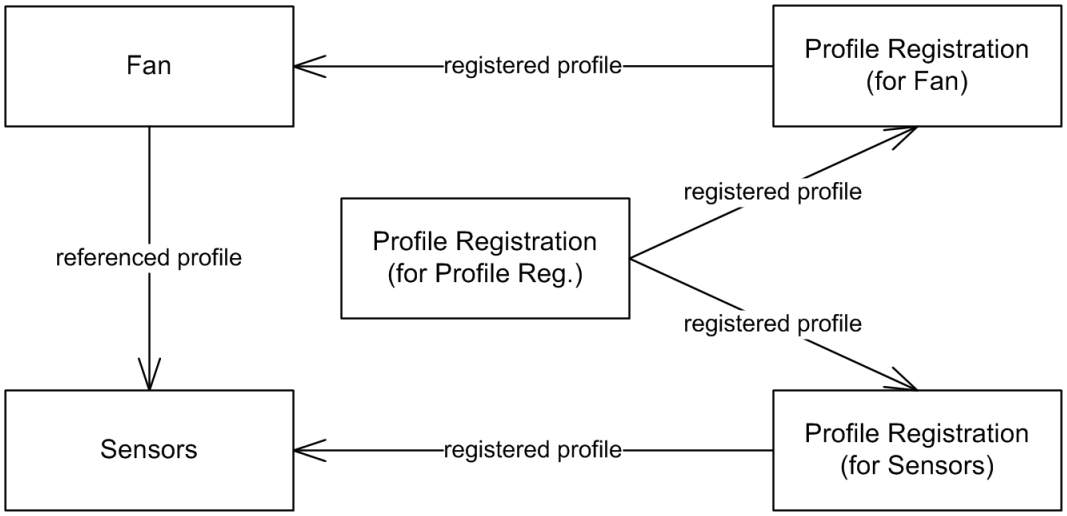


Figure 1 – Profile relationships example

The Profile Registration profile itself is also a registered profile and is therefore advertised through its Profile Registration profile (another implementation of the same profile). This is shown in Figure 2:

152



153 **Figure 2 – Profile relationships example with Profile Registration advertisement**

154

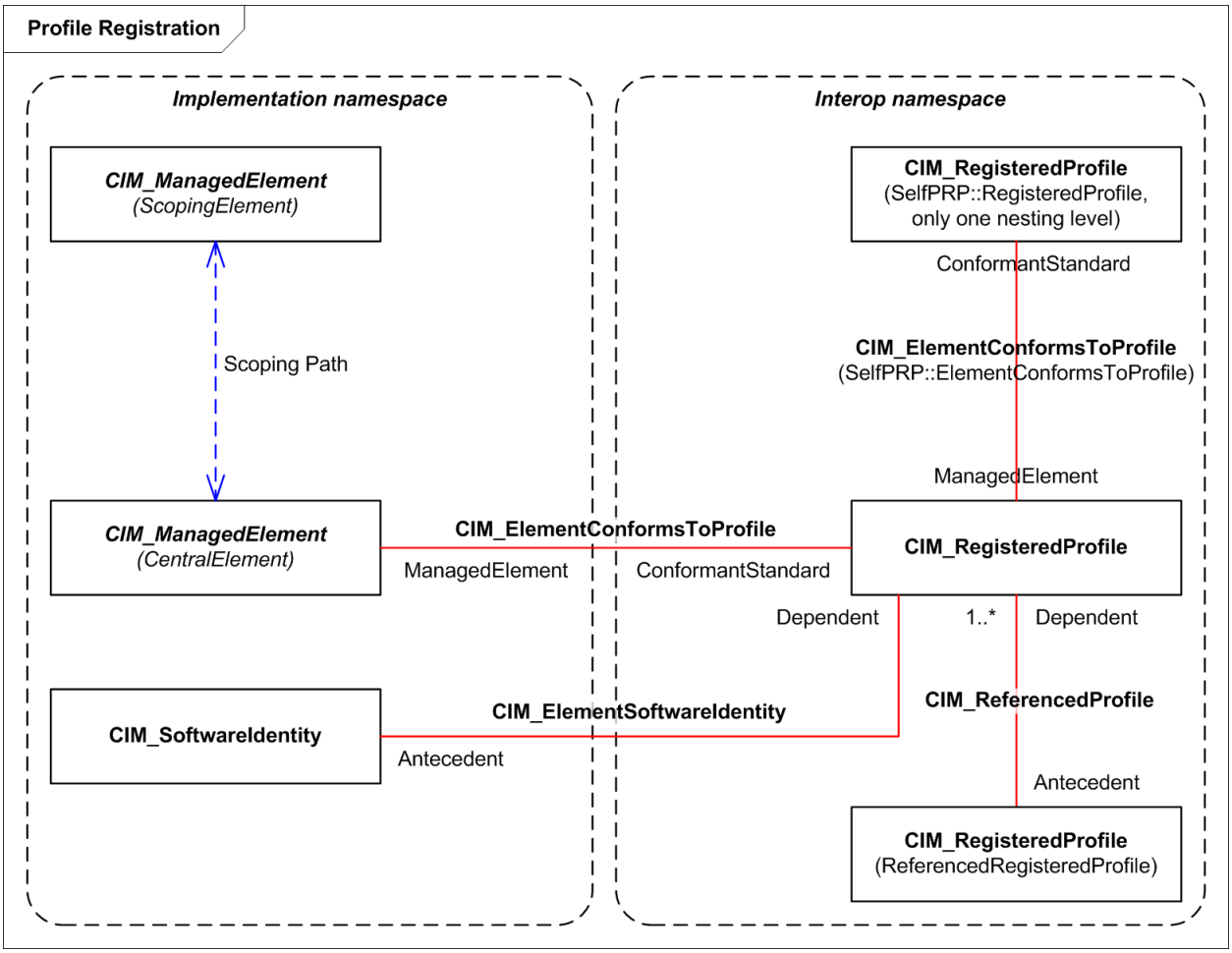
For simplicity, these two figures have left out that each of the Fan, Sensors and Profile Registration profiles also references the Profile Registration profile.

155

6.2 DMTF adaptation class diagram

156 Figure 3 shows all class adaptations defined in this profile, and relevant class adaptations from referenced profiles. Adaptation names are shown in parentheses below the class names if they differ from the class names without schema prefix.

157



158 **Figure 3 – DMTF adaptation class diagram**

159

Registered profiles (that is, profiles to which an implementation advertises conformance) are represented by instances of the RegisteredProfile adaptation in the Interop namespace.

160 As defined in 6.4, the roles of an Interop namespace and of an implementation namespace can be assumed by different namespaces or by the same namespace. Figure 3 shows the case of different namespaces. If these namespaces are different, the class adaptations shown in the Interop namespace may also be implemented in the implementation namespace (that is, they appear in both namespaces).

161 The RegisteredProfile class adaptation is the central and scoping class adaptation of this profile.

162 The central and scoping elements of the registered profile are represented by instances of the CentralElement and ScopingElement adaptation, respectively.

163 If the ElementConformsToProfile adaptation is implemented, the registered profile supports the central class methodology; the scoping class methodology is always supported. For a complete definition, see 6.3.

164 If the registered profile references any profiles, these referenced profiles are represented by instances of the ReferencedRegisteredProfile class adaptation. These instances are associated via the ReferencedProfile association adaptation to the instances of the RegisteredProfile class adaptation that represent the referencing profile.

165 The referenced profiles also advertise their profile conformance through this profile.

166

If the registered profile is a component profile, it has a scoping profile. Conformance of an implementation to the scoping profile is also advertised through a use of this profile. This configuration is not shown in the diagram; the diagram only shows how this profile is used by the registered profile. A use of this profile for advertising conformance of an implementation to the scoping profile results from the fact that the scoping profile references this profile as well, so it is on the role of a registered profile and the diagram is simply applied another time using that role.

167 An implementation that uses this profile to advertize a registered profile has implemented this profile and thus also needs to advertize conformance to this profile. In other words, this profile takes on the role of a registered profile for this purpose. The resulting profile reference is named "SelfPRP" in Table 1; and that use of this profile is shown in Figure 3 in the adaptations "SelfPRP::RegisteredProfile" and "SelfPRP::ElementConformsToProfile" . Conceptually, that advertisement is again an implementation of this profile, but in order to avoid nesting this concept at arbitrary depth, it has been limited to be nested only one level deep, so that the RegisteredProfile instance representing conformance to this profile is not subject to further advertisement.

168 The SoftwareIdentity and ElementSoftwareIdentity adaptations provide support for representing the software identity of the implementation that conforms to the registered profile; they are part of the SoftwareIdentity feature.

169

6.3 Central and scoping class concept

170

6.3.1 General

171 Profiles typically define constraints and behavioral requirements for more than one CIM schema class. The usages of CIM schema classes in the context of a profile are termed *adaptations* (see DSP1001). For an implementation to conform to a profile, each of the CIM elements for which the profile defines constraints and behavioral requirements needs to conform to these constraints and behavioral requirements. Because profiles also define which entities in the managed environment are represented by the model entities, conformance to a profile cannot only be limited to *interface conformance* (see DSP1001), but needs to include those mapping aspects as well. Therefore, an implementation conforms to a profile, if it satisfies the rules for *full implementation conformance* defined in 5.2.2 of DSP1001.

172 This profile establishes the concepts of a *central class adaptation* and a *scoping class adaptation* that allow a client to perform the following tasks:

173 • to find the CIM instances that conform to the registered profile, given the RegisteredProfile instance representing the registered profile

174 • to find - for a given CIM instance - the RegisteredProfile instance (or instances) representing the registered profile (or profiles), to which conformance is advertised

175 The *central class adaptation* of a profile acts as an algorithmic focal point for all adaptations defined by that profile. The central class adaptation also represents the boundary for clients between using a generic discovery mechanism and using a priori knowledge about the profile, as follows:

176 • Navigation between the RegisteredProfile instance representing a registered profile and its central instances is defined in this profile with generic discovery mechanisms called *profile advertisement methodologies*; some of these do not require clients to have a priori knowledge about the particular profile.

177 • Traversal between the central instances of a registered profile and the instances of adaptations defined by that profile requires clients to have a priori knowledge about the profile; this profile does not define generic discovery mechanisms for that purpose.

178 Implementations that conform to multiple profiles and implementations that conform to profiles and in addition implement schema classes outside of the context of any profile deserve particular attention by

clients, when navigating the network of instances, because it is possible that instances of a particular class conform to different profiles or to no profile. This often requires clients to have a priori knowledge about the way these multiple profiles and schema classes have been combined in the implementation.

179 The *scoping class adaptation* of a profile is used for discovering the central instances indirectly, in cases where there are many central instances to be expected.

180 In autonomous profiles, the central class adaptation and the scoping class adaptation are the same adaptation (see DSP1001), with the same set of instances.

181 This profile defines three profile advertisement methodologies through which an implementation can advertise conformance to a particular profile, and through which clients can navigate between the RegisteredProfile instance representing the registered profile and its central instances:

183 • The first methodology is termed *central class methodology*; it is characterized by a direct ElementConformsToProfile association adaptation between the CentralElement and RegisteredProfile adaptations. This means, every central instance is directly associated with the RegisteredProfile instance representing the registered profile.

184 See 6.3.2 for more information about the central class methodology.

186 • The second methodology is termed *scoping class methodology*; it uses the ElementConformsToProfile association adaptation only between the ScopingElement adaptation of the registered profile and the RegisteredProfile adaptation of the scoping profile.

187 The ScopingElement adaptation of the registered profile binds to the CentralElement adaptation of the scoping profile, so this profile advertisement methodology basically delegates the traversal of the ElementConformsToProfile association adaptation to the scoping profile.

188 This delegation may happen across multiple levels of scoping profiles, until some scoping profile finally implements the central class methodology. It is typical (but not required) that that final scoping profile is an autonomous profile.

189 See 6.3.3 for more information about the scoping class methodology.

191 • The third methodology is termed *GetCentralInstances methodology*; it is characterized by a method GetCentralInstances( ) defined in RegisteredProfile that returns the central instances directly. This approach is very efficient because the implementation typically knows its central instances.

192 See 6.3.2 for more information about the central class methodology.

193 The scoping class methodology is always implemented and available for use.

194 The central class methodology may be implemented in addition (see feature CentralClassMethodology). The decision about implementing the central class methodology should be left to the implementation; that is, profiles should not normally require or prohibit this methodology to be implemented.

195 The GetCentralInstances methodology may be implemented in addition (see feature GetCentralInstancesMethodology). The decision about implementing the GetCentralInstances methodology should be left to the implementation; that is, profiles should not normally require or prohibit this methodology to be implemented.

196 For autonomous profiles, the scoping class methodology effectively becomes the same as the central class methodology, because scoping element and central element are the same.

197 In situations where implementations have small footprint requirements and want to reduce the number of instances or in situations where the implementation is monolithic and only a single version of each profile is used, the implementation may use the scoping class methodology (by not implementing the central class methodology) to reduce the number of necessary ElementConformsToProfile instances.

198

In situations where WBEM servers support multiple implementations of the same or different versions of a profile, the central class methodology is recommended, because it provides unambiguous relationships through ElementConformsToProfile instances between central instances and the RegisteredProfile instances representing the registered profiles with their versions.

199 If such multiple implementations of the same or different versions of a profile make different decisions for implementing the central class methodology, that can result in limitations for discovering the central instances. For example, a client will find the central instances of those profile implementations that used the central class methodology, but has no way to determine whether this is the complete list of central instances (except for trying the central class methodology in addition).

200 An example of this scenario could be a system with two network interface cards, each from a different vendor, and the parts of the overall implementation contributed by each vendor conform to different versions of the Ethernet Port Profile. This scenario also shows that in multi-vendor environments, it may be difficult to coordinate the choice of profile advertisement methodology. Using the central class methodology puts a profile implementation on the safe side in multi-vendor environments.

201

6.3.2 Central class methodology

202 The central class profile advertisement methodology (or short: central class methodology) is based on a straightforward approach whereby every CentralElement instance (representing the central instances of a registered profile) is associated through ElementConformsToProfile with a RegisteredProfile instance that represents the registered profile and version to which the profile implementation advertises conformance.

203 This profile advertisement methodology is straightforward because clients only need to traverse the ElementConformsToProfile association adaptation from or to the profile's CentralElement instance to ascertain the profiles to which the implementation advertises conformance.

204 Using this profile advertisement methodology is covered by the CentralClassMethodology feature.

205 Figure 4 is an object diagram (showing unnamed instances) that provides an example of the central class methodology of advertising profile conformance. In the figure, the dotted line bi-directional arrows represent the ability of a client to traverse the ElementConformsToProfile association adaptation in the following ways:

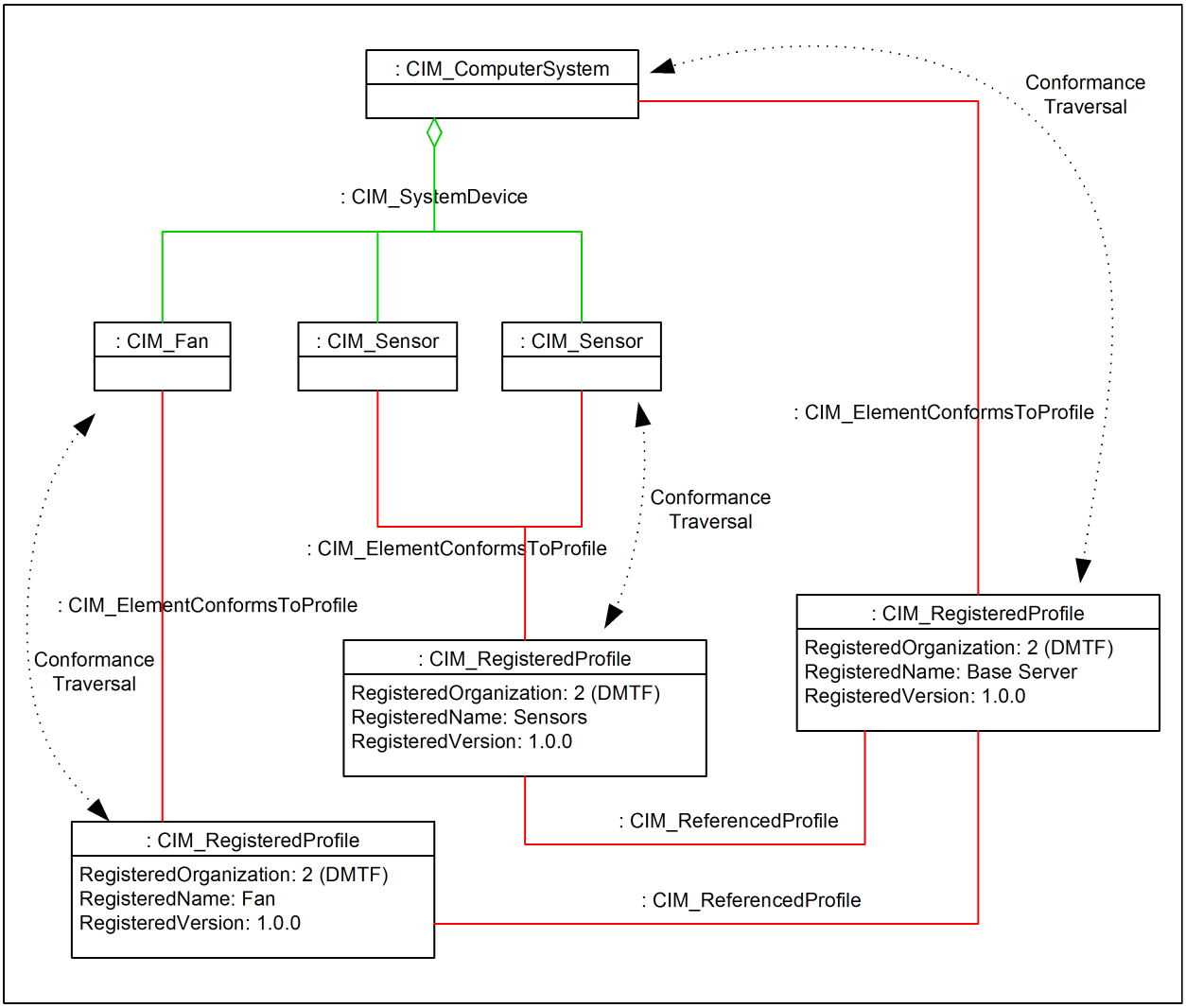
206 • from a central instance of the registered profile to the RegisteredProfile instance that represents that profile. Note that a particular CIM instance can act as a central instance for more than one profile.

207 • from a RegisteredProfile instance that represents a registered profile to the central instances of that profile.

208 In both cases, the traversal of the ElementConformsToProfile adaptation typically will be across namespaces; that is not represented in Figure 4 but is described in 6.4.4.

209 In Figure 4, the ComputerSystem, Fan, and Sensor adaptations are defined in respective profiles; they are all central elements in these profiles and are therefore based on the CentralElement adaptation defined in this profile. The RegisteredProfile instances represent these three profiles. It is furthermore assumed that for the purposes of this example, that the Sensors profile is implemented for some system level sensor (and not for a fan sensor).

210



211 **Figure 4 – Central class methodology example**

212

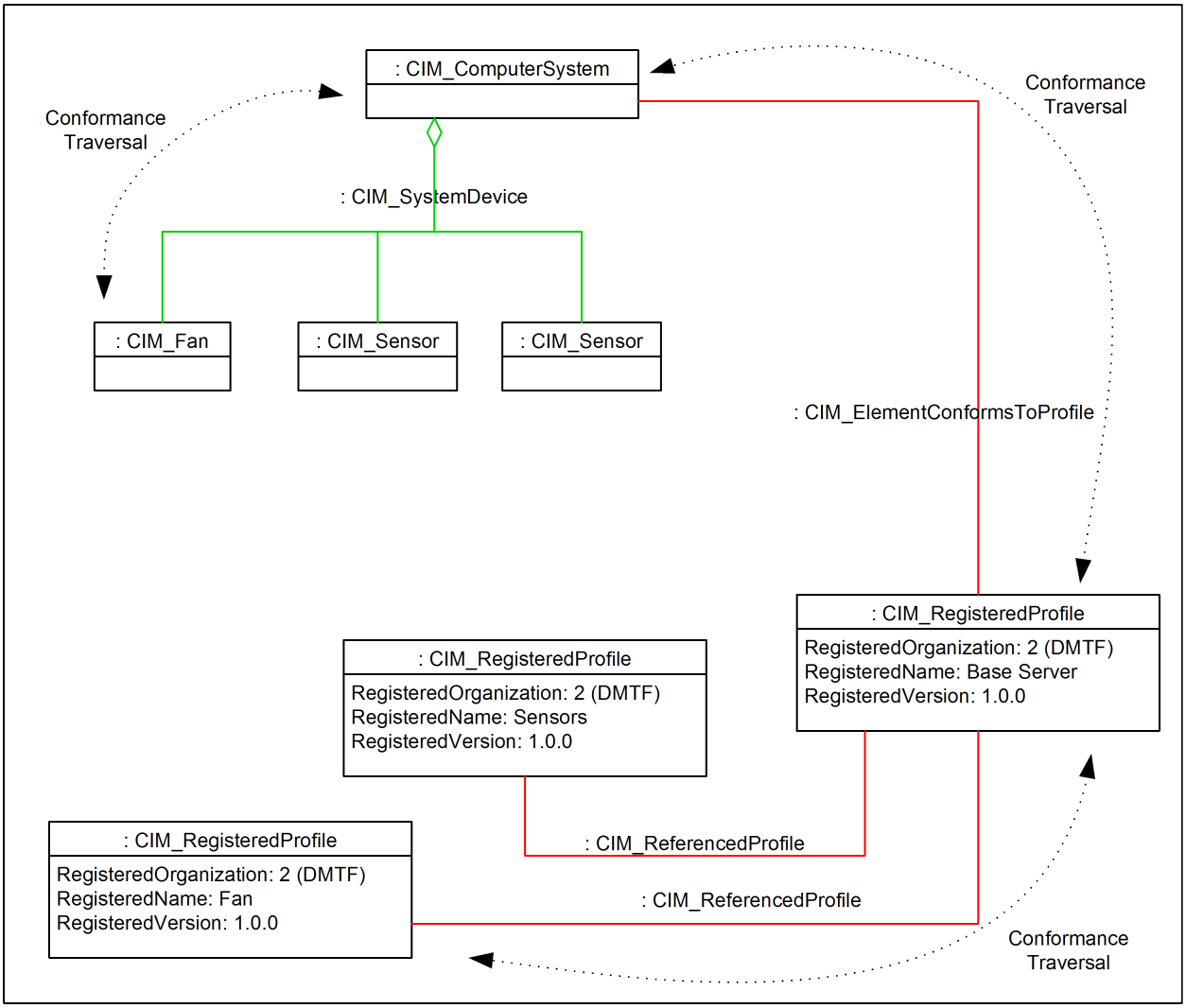
6.3.3 Scoping class methodology

213 The scoping class profile advertisement methodology (or short: scoping class methodology) is an approach characterized by the use of the ElementConformsToProfile association adaptation not between the central instances of a registered profile and a RegisteredProfile instance that represents that registered profile, but instead by having that association adaptation at the next scoping profile that uses the central class methodology for itself.

214 This profile advertisement methodology is always implemented and available for use (that is, even when the central class methodology is implemented in addition).

215 Figure 5 is an object diagram (showing unnamed instances) that provides an example of the scoping class methodology of advertising profile conformance with one level of scoping profiles.

216



217 **Figure 5 – Scoping class methodology example**

218

In Figure 5, a client may traverse from a Fan instance to its scoping instance (the ComputerSystem instance) through the SystemDevice association adaptation, following the scoping path defined in the Example Fan profile. Because the ComputerSystem instance is referenced by ElementConformsToProfile instances, the client knows that the corresponding profile has used the central class methodology, and can now traverse ElementConformsToProfile to a RegisteredProfile instance that represents the Example Base Server profile, version 1.0.0, which is the scoping profile of the Example Fan profile. Finally, ReferencedProfile is traversed to a RegisteredProfile instance that represents the Example Fan profile, version 1.0.0, to which the implementation is advertising conformance.

219 The client may reverse this traversal and start from the RegisteredProfile instance that represents the Example Fan profile to get to the instance(s) of Fan.

220 The concept is in both cases that the client navigates up the scoping profile hierarchy to the level where a scoping profile uses the central class methodology (as indicated by the presence of instances of the ElementConformsToProfile association adaptation), and then traverses from the element side to the profile side or vice versa, and then navigates down the scoping profile hierarchy the same number of steps.

221

In both cases, the traversal of the ElementConformsToProfile adaptation typically will be across namespaces; that is not represented in Figure 5 but is described in 6.4.4.

222 In Figure 5, the ComputerSystem, Fan, and Sensor adaptations are defined in respective profiles; they are all central elements in these profiles and are therefore implicitly based on the CentralElement adaptation defined in this profile. The RegisteredProfile instances represent these three profiles.

223

6.3.4 GetCentralInstances methodology

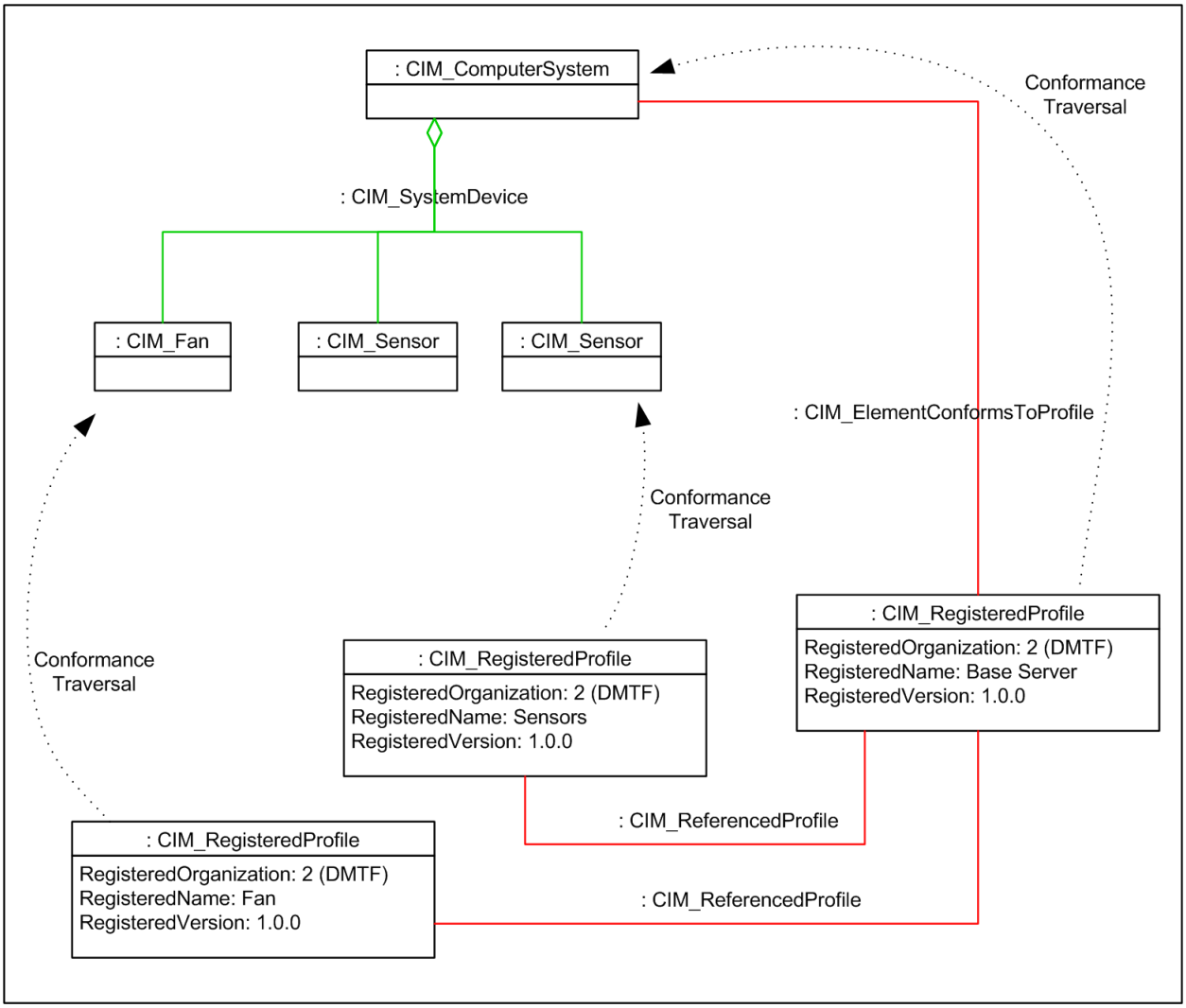
224 The GetCentralInstances methodology uses the GetCentralInstances( ) method on a RegisteredProfile instance to return the central instances of the profile advertised by that instance.

225 The ElementConformsToProfile association does not need to be implemented for this methodology to work.

226 However, this methodology only allows determining the central instances from the RegisteredProfile instance, but not vice versa.

227 Figure 4 is an object diagram (showing unnamed instances) that provides an example of the GetCentralInstances methodology. In the figure, the dotted line uni-directional arrows represent the ability of a client to determine the central instances from the RegisteredProfile instance.

228



229 **Figure 6 – GetCentralInstances methodology example**

230

6.4 WBEM server requirements on CIM namespaces

231 This subclause defines the roles of Interop namespace and implementation namespace for CIM namespaces, and related implementation requirements for WBEM servers.

232 Some of these concepts and requirements have a more general scope than this profile. For example, the concept of an Interop namespace is also used by other profiles (e.g., DSP1054) or by WBEM SLP discovery (see DSP0206). Another such example is the concept of cross-namespace associations.

233

6.4.1 Interop namespace

234 *Interop namespace* is a role of a CIM namespace for the purpose of providing a common and well-known place for clients to discover modeled entities, such as the profiles to which an implementation advertises conformance.

235 A WBEM server shall implement one CIM namespace and may implement additional CIM namespaces that assume the role of an Interop namespace; each of these namespaces is termed an Interop namespace.

236

At least one Interop namespace of a WBEM server shall have one of the following standard names:

- `interop` (preferred)
- `/interop` **(DEPRECATED)**
- `root/interop` **(DEPRECATED)**
- `/root/interop` **(DEPRECATED)**

Clients need to be prepared to deal with any one of these standard names for the Interop namespace.

A WBEM server may expose Interop namespaces using additional implementation-defined names. This accommodates backwards compatibility of existing WBEM server implementations. Clients should use the standard names instead of such implementation-defined names.

If a WBEM server implements multiple Interop namespaces (using standard names or implementation-defined names), each of those namespaces shall expose a distinct set of CIM instances (that is, instances with a different namespace path), that represent equivalent information (that is, their property values are the same except for different namespace paths in references).

DEPRECATED

The use of `root/interop` for the Interop namespace name has been deprecated in version 1.1 of this profile.

DEPRECATED

DEPRECATED

The use of `/interop` and `/root/interop` for the Interop namespace name, and more generally the use of leading slash (`/`) characters in any namespace name have been deprecated in version 1.1 of this profile. Older WBEM implementations may have considered the slash separator character in a CIM object path URI to be part of the namespace name and thus exposed the namespace name (e.g., in the Name property of CIM_Namespace) with a leading slash character. DSP0004 does not permit namespace names to begin with a slash.

Producers of Interop namespace names should not create a leading slash (`/`) character in the Interop namespace name. Consumers of Interop namespace names shall ignore a leading slash character in Interop namespace names when processing them (e.g., for comparison or identification purposes).

DEPRECATED

6.4.2 Implementation namespaces

Implementation namespace is a role of a CIM namespace for the purpose of providing a place for CIM objects for which no specific namespace requirements are defined.

A WBEM server shall implement one or more CIM namespaces that assume the role of an implementation namespace; each such namespace is also called an implementation namespace.

The names of implementation namespaces are implementation-defined.

6.4.3 Relationship between Interop and implementation namespaces

A CIM namespace of a WBEM server may play the roles of an implementation namespace and of an Interop namespace at the same time.

Thus, a simple implementation of a WBEM server can expose a single CIM namespace that plays both roles. Of course, that single CIM namespace needs to satisfy the requirements for its name as defined in 6.4.1.

258 A typical implementation of a WBEM server will expose a single Interop namespace and multiple implementation namespaces, each of which is a distinct namespace implementation.

259 The part of an implementation that conforms to a particular single profile may span multiple namespaces, including multiple implementation namespaces.

260

6.4.4 Cross-namespace associations

261 Some association adaptations defined in this profile may cross CIM namespaces (within the same WBEM server).

262 Associations that cross CIM namespaces shall be instantiated in both namespaces. The rationale for this is to support association traversal from either namespace to the other.

263 Each of these association instances shall have their creation class exist in the same namespace as the association instance. The versions of these association classes in each of the two namespaces may be different; this is needed in order to allow that the implementation namespaces within a WBEM server can be used for objects from different versions of the CIM schema.

264

7 Implementation

265

7.1 Features

266

7.1.1 Feature: CentralClassMethodology

267 **Requirement level:** Optional

268 Implementing this feature for a registered profile provides support for advertising conformance of an implementation to that registered profile using the central class methodology. For details, see 6.3.2.

269 This feature shall be implemented for autonomous profiles. Note that the Profile Registration profile (this profile) is an autonomous profile.

270 Note that the scoping class methodology is always implemented and available for use.

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

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

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

275 OCL context: A RegisteredProfile instance.

276
```
derive: self->CIM_ElementConformsToProfile->size() > 0
```

277 Explanation:

278 At least one ElementConformsToProfile instance exists that references the RegisteredProfile instance representing the registered profile.

279 This discovery mechanism only works if at least one central instance exists and if all implementations of the registered profile in a particular WBEM server use the same methodology.

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

281

7.1.2 Feature: GetCentralInstancesMethodology

Requirement level: Optional

Implementing this feature for a registered profile provides support for advertising conformance of an implementation to that registered profile using the GetCentralInstances( ) method. For details, see 6.3.4.

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

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

7.1.3 Feature: SoftwareIdentity

Requirement level: Optional

Implementing this feature for a registered profile provides support for representing the software identity of an implementation that conforms to that profile. That software identity is represented using the SoftwareIdentity adaptation which is associated to the RegisteredProfile adaptation representing conformance to the registered profile via the ElementSoftwareIdentity adaptation.

A particular SoftwareIdentity instance represents the software identity of one implementation and can be related to one or more registered profiles.

A particular registered profile can have more than one software identity, each represented by a SoftwareIdentity instance. For example, this can happen if the core functionality of a profile is in one implementation, and a second implementation adds support for an optional feature of that profile.

The SoftwareIdentity and ElementSoftwareIdentity adaptations defined in this profile have been designed to conform to the CIM_SoftwareIdentity and CIM_ElementSoftwareIdentity classes, respectively, that are used in the Software Inventory Profile (DSP1023).

Nevertheless, the Software Identity Profile is not referenced by this profile for several reasons:

- the Software Identity Profile defines CIM_System as its scoping class, but this profile is an autonomous profile that does not define CIM_System
- the reference circle between the Software Inventory Profile and this profile would have been complex to handle, particularly considering the usage of this profile by itself

The disadvantage of this approach is that the conformance of this feature to the Software Identity Profile cannot be discovered by clients. However, it is possible to reuse CIM_SoftwareIdentity instances that are implemented as part of the Software Inventory Profile also for this profile. If that is done, note that the SoftwareIdentity and ElementSoftwareIdentity adaptations define constraints in addition to the CIM_SoftwareIdentity and CIM_ElementSoftwareIdentity classes that are used in the Software Inventory Profile.

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

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

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

  OCL context: A RegisteredProfile instance.

  ```
  derive: self->CIM_ElementSoftwareIdentity->size() > 0
  ```

  Explanation:

  A SoftwareIdentity instance exists that is associated to the RegisteredProfile instance via the ElementSoftwareIdentity association.

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

7.2 Adaptations

7.2.1 Conventions

This profile defines operation requirements based on DSP0223.

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

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

The default initialization requirement level for property requirements is optional.

The default modification requirement level for property requirements is optional.

This profile repeats the effective values of certain Boolean qualifiers as part of property, method parameter, or method return value requirements. The following convention is established: If the name of a qualifier is listed, its effective value is True; if the qualifier name is not listed, its effective value is False. The convention is applied in the following cases:

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

7.2.2 Adaptation: RegisteredProfile: CIM_RegisteredProfile

7.2.2.1 General

Adaptation type: Ordinary class

Implementation type: Instantiated

Requirement level: Mandatory

This adaptation models registered profiles (that is, profiles to which an implementation advertises conformance).

It is important to understand that this adaptation does not model "profile implementations" that could be distinguished within an overall implementation. The overall implementation may be a mix of components from different vendors, each of which may have implemented a profile, but these different parts are not necessarily distinguishable within the overall implementation. Only the conformance of the overall implementation to a profile is modeled with this adaptation.

Table 5 – RegisteredProfile: Element requirements

Element	Requirement	Description
Properties		
InstanceID	Mandatory	Key

Element	Requirement	Description
RegisteredOrganization	Mandatory	Required
RegisteredName	Mandatory	Required
RegisteredVersion	Mandatory	Required
AdvertiseTypes	Mandatory	Required
OtherRegisteredOrganization	Conditional	See 7.2.2.2
AdvertiseTypeDescriptions	Conditional	See 7.2.2.3
SpecificationType	Mandatory	See 7.2.2.4
ImplementedFeatures	Mandatory	
Methods		
GetCentralInstances( )	Conditional	See 7.2.2.5
Operations		
GetInstance( )	Mandatory	
EnumerateInstances( )	Mandatory	
EnumerateInstanceNames( )	Mandatory	
OpenEnumerateInstances( )	Optional	
Associators( )	Mandatory	
AssociatorNames( )	Mandatory	
OpenAssociators( )	Optional	
References( )	Mandatory	
ReferenceNames( )	Mandatory	
OpenReferences( )	Optional	

7.2.2.2 Property: OtherRegisteredOrganization

Requirement level: Conditional

Condition:

The following OCL statement evaluates to true in the context of a RegisteredProfile instance:

```
self.RegisteredOrganization = 1 /* Other */
```

7.2.2.3 Property: AdvertiseTypeDescriptions

Requirement level: Conditional

Condition:

The following OCL statement evaluates to true in the context of a RegisteredProfile instance:

```
self.AdvertiseTypes->exists( value | value = 1 /* Other */ )
```

Explanation:

The AdvertiseTypes array property has at least one array entry with a value of 1 (Other).

Constraint:

OCL constraint in the context of a RegisteredProfile instance:

```
inv: Sequence { 1 .. self.AdvertiseTypes->size() }->
    forAll( i |
        self.AdvertiseTypes.at(i) = 1 /* Other */
        implies self.AdvertiseTypeDescriptions.at(i) != null
    )
```

Explanation:

For each array entry of AdvertiseTypes that has a value of 1 (Other), the corresponding array entry of AdvertiseTypeDescriptions shall be non-Null.

Note that this constraint leaves the value of array entries of AdvertiseTypeDescriptions undefined, including the possibility of being Null or not present (after any non-Null array entries). As a result, if no array entry of AdvertiseTypes has a value of 1 (Other), the AdvertiseTypeDescriptions property is entirely undefined, including the possibility of it being Null.

7.2.2.4 Property: SpecificationType

Requirement level: Mandatory

Constraint:

OCL constraint in the context of a RegisteredProfile instance:

```
inv: self.SpecificationType = 2 /* Profile */
```

7.2.2.5 Method: GetCentralInstances( )

Requirement level: Conditional

Condition:

The GetCentralInstancesMethodology feature is implemented.

Table 6 – GetCentralInstances( ): Parameter requirements

Parameter	Description
CentralInstances	Out, see 7.2.2.5.1

7.2.2.5.1 Parameter: CentralInstances

Constraint:

Referenced instances shall be of class adaptation CentralElement.

7.2.3 Adaptation: ElementConformsToProfile: CIM_ElementConformsToProfile

7.2.3.1 General

Adaptation type: Association class

Implementation type: Instantiated

Requirement level: Conditional exclusive

Condition:

The CentralClassMethodology feature is implemented.

364 Note that if the CentralClassMethodology feature is not implemented, traversal between RegisteredProfile and CentralElement instances is delegated to the level of the scoping profile, as described in 6.3.

365 This adaptation models the relationship between registered profiles and their central instances.

366 **Table 7 – ElementConformsToProfile: Element requirements**

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

367 #### 7.2.3.2 Property: ConformantStandard

368 **Requirement level:** Mandatory

369 **Reference kind:** REF-typed

370 **Constraint:**

371 Referenced instances shall be of class adaptation RegisteredProfile.

The multiplicity of this association end is 0 .. *

372 #### 7.2.3.3 Property: ManagedElement

373 **Requirement level:** Mandatory

374 **Reference kind:** REF-typed

375 **Constraint:**

376 Referenced instances shall be of class adaptation CentralElement.

The multiplicity of this association end is 0 .. *

377 ### 7.2.4 Adaptation: ScopingElement: CIM_ManagedElement

378 This adaptation models scoping elements of registered profiles.

379 This adaptation shall be (implicitly) applied as a base adaptation to the scoping class adaptation of the registered profile; that is, that adaptation does not need to specify this adaptation is its base adaptation, but is still considered a derived adaptation of this adaptation.

380 **Adaptation type:** Ordinary class

381 **Implementation type:** Abstract

382 **Requirement level:** Defined by its derived adaptations

383

7.2.5 Adaptation: CentralElement: CIM_ManagedElement

384 This adaptation models central elements of registered profiles. Note that DSP1001 requires that every DMTF profile references this profile, and requires that referencing profiles base their central class adaptation on this adaptation.

385 This adaptation shall be (implicitly) applied as a base adaptation to the central class adaptation of the registered profile; that is, that adaptation does not need to specify this adaptation is its base adaptation, but is still considered a derived adaptation of this adaptation.

386 **Adaptation type:** Ordinary class

387 **Implementation type:** Abstract

388 **Requirement level:** Defined by its derived adaptations

389 **Table 8 – CentralElement: Element requirements**

Element	Requirement	Description
Operations		
Associators( )	Mandatory	
AssociatorNames( )	Mandatory	
OpenAssociators( )	Optional	
References( )	Mandatory	
ReferenceNames( )	Mandatory	
OpenReferences( )	Optional	

390

7.2.6 Adaptation: ReferencedProfile: CIM_ReferencedProfile

391

7.2.6.1 General

392 **Adaptation type:** Association class

393 **Implementation type:** Instantiated

394 **Requirement level:** Mandatory

395 This adaptation models the relationship between registered profiles and the profiles they reference and for which conformance is advertised.

396 **Table 9 – ReferencedProfile: Element requirements**

Element	Requirement	Description
Properties		
Antecedent	Mandatory	Key, see 7.2.6.2
Dependent	Mandatory	Key, see 7.2.6.3
Operations		
GetInstance( )	Mandatory	
EnumerateInstances( )	Mandatory	
EnumerateInstanceNames( )	Mandatory	
OpenEnumerateInstances( )	Optional	

397

7.2.6.2 Property: Antecedent

Requirement level: Mandatory

Reference kind: REF-typed

Constraint:

Referenced instances shall be of class adaptation ReferencedRegisteredProfile.

The multiplicity of this association end is 0 .. *

7.2.6.3 Property: Dependent

Requirement level: Mandatory

Reference kind: REF-typed

Constraint:

Referenced instances shall be of class adaptation RegisteredProfile.

The multiplicity of this association end is 0 .. *

7.2.7 Adaptation: ReferencedRegisteredProfile: CIM_RegisteredProfile

This adaptation models referenced profiles; that is, profiles that are referenced by the registered profile (represented by the RegisteredProfile adaptation instance) and for which conformance is advertised. The type of profile relationship can be "usage" or "derivation" (see DSP1001).

This adaptation and the ReferencedProfile adaptation together provide the ability to navigate the relationships between profiles that are advertised. However, the type of relationship is not represented.

Note that such referenced registered profiles are also considered normal registered profiles in the context of the referenced profile. That is expressed by the base adaptation RegisteredProfile in the referenced profile (see the RefPRP profile reference).

Adaptation type: Ordinary class

Implementation type: Instantiated

Requirement level: Mandatory

Table 10 – ReferencedRegisteredProfile: Element requirements

Element	Requirement	Description
Base adaptations		
RefPRP::RegisteredProfile	Mandatory	See RefPRP::RegisteredProfile.
Operations		
Associators( )	Mandatory	
AssociatorNames( )	Mandatory	
OpenAssociators( )	Optional	
References( )	Mandatory	
ReferenceNames( )	Mandatory	
OpenReferences( )	Optional	

7.2.8 Adaptation: SoftwareIdentity: CIM_SoftwareIdentity

7.2.8.1 General

Adaptation type: Ordinary class

Implementation type: Instantiated

Requirement level: Conditional

Condition:

The SoftwareIdentity feature is implemented.

This adaptation models the software identity of implementations that conform to the registered profiles represented by RegisteredProfile instances associated via ElementSoftwareIdentity.

Note that this adaptation has been designed to conform to the CIM_SoftwareIdentity class used in DSP1023.

The algorithm for version comparison using the MajorVersion, MinorVersion, RevisionNumber, and BuildNumber properties defined in DSP1023 shall be used for comparing versions of software identities represented by instances of this adaptation.

Table 11 – SoftwareIdentity: Element requirements

Element	Requirement	Description
Properties		
InstanceID	Mandatory	Key
IsEntity	Mandatory	
VersionString	Optional	
MajorVersion	Conditional	See 7.2.8.2
MinorVersion	Conditional	See 7.2.8.3
RevisionNumber	Conditional	See 7.2.8.4
BuildNumber	Conditional	See 7.2.8.5
Operations		
GetInstance( )	Mandatory	
EnumerateInstances( )	Optional	
EnumerateInstanceNames( )	Mandatory	
OpenEnumerateInstances( )	Optional	
Associators( )	Mandatory	
AssociatorNames( )	Mandatory	
OpenAssociators( )	Optional	
References( )	Mandatory	
ReferenceNames( )	Mandatory	
OpenReferences( )	Optional	

7.2.8.2 Property: MajorVersion

Requirement level: Conditional

Condition:

The following OCL statement evaluates to true in the context of a SoftwareIdentity instance:

```
self.VersionString = null
```

7.2.8.3 Property: MinorVersion

Requirement level: Conditional

Condition:

The following OCL statement evaluates to true in the context of a SoftwareIdentity instance:

```
self.VersionString = null
```

7.2.8.4 Property: RevisionNumber

Requirement level: Conditional

Condition:

The following OCL statement evaluates to true in the context of a SoftwareIdentity instance:

```
self.VersionString = null
```

7.2.8.5 Property: BuildNumber

Requirement level: Conditional

Condition:

The following OCL statement evaluates to true in the context of a SoftwareIdentity instance:

```
self.VersionString = null
```

7.2.9 Adaptation: ElementSoftwareIdentity: CIM_ElementSoftwareIdentity

7.2.9.1 General

Adaptation type: Association class

Implementation type: Instantiated

Requirement level: Conditional

Condition:

The SoftwareIdentity feature is implemented.

This adaptation models the relationship between registered profiles and the software identity of their implementation.

Note that this adaptation has been designed to conform to the CIM_ElementSoftwareIdentity class used in DSP1023.

Table 12 – ElementSoftwareIdentity: Element requirements

Element	Requirement	Description
Properties		
Antecedent	Mandatory	Key, see 7.2.9.2
Dependent	Mandatory	Key, see 7.2.9.3
ElementSoftwareStatus	Mandatory	See 7.2.9.4

Element	Requirement	Description
Operations		
GetInstance( )	Mandatory	
EnumerateInstances( )	Mandatory	
EnumerateInstanceNames( )	Mandatory	
OpenEnumerateInstances( )	Optional	

7.2.9.2 Property: Antecedent

Requirement level: Mandatory

Reference kind: REF-typed

Constraint:

Referenced instances shall be of class adaptation SoftwareIdentity.

The multiplicity of this association end is 0 .. *

7.2.9.3 Property: Dependent

Requirement level: Mandatory

Reference kind: REF-typed

Constraint:

Referenced instances shall be of class adaptation RegisteredProfile.

The multiplicity of this association end is 1 .. *

7.2.9.4 Property: ElementSoftwareStatus

Requirement level: Mandatory

Constraint:

OCL constraint in the context of a ElementSoftwareIdentity instance:

```
inv: self.ElementSoftwareStatus = Set { 2 /* Current */, 6 /* Installed */ }
```

Explanation:

The ElementSoftwareStatus array property shall contain the values 2 (Current) and 6 (Installed), in any order.

8 Use cases and state descriptions

8.1 State description: SimpleStateDescription

This state description describes a simple scenario in which an implementation conforms to three example profiles, and advertises conformance through this profile (i.e., the Profile Registration profile). In this state description, each implementation of this profile in turn advertises conformance to this profile itself.

Table 13 lists these four profiles, and their referenced profiles:

Table 13 – Profiles in the SimpleStateDescription scenario

Profile	Profile Type	Referenced Profile	Profile Reference Type	Profile Reference Name
Example Base Server	Autonomous	Profile Registration	Usage	PRP
		Example Fan	Usage	SystemFan
		Example Power Supply	Usage	SystemPowerSupply
Example Fan	Component	Profile Registration	Usage	PRP
Example Power Supply	Component	Profile Registration	Usage	PRP
Profile Registration	Autonomous	Profile Registration	Usage	SelfPRP
		Profile Registration	Usage	RefPRP

482 Table 14 lists the class adaptations defined in the three example profiles and in this profile, to the extent they are relevant for this scenario.

483 **Table 14 – Adaptations in the SimpleStateDescription scenario**

Profile	Adaptation	Schema Class	Base Adaptation	Profile Reference Name (of Base Adaptation)
Example Base Server	ComputerSystem (central + scoping element)	CIM_ComputerSystem	ScopingElement (implied)	PRP
			CentralElement (implied)	PRP
			System	SystemFan
			System	SystemPowerSupply
Example Fan	System (scoping element)	CIM_System	ScopingElement (implied)	PRP
	SystemDevice	CIM_SystemDevice		
	Fan (central element)	CIM_Fan	CentralElement (implied)	PRP
Example Power Supply	System (scoping element)	CIM_System	ScopingElement (implied)	PRP
	SystemDevice	CIM_SystemDevice		
	PowerSupply (central element)	CIM_PowerSupply	CentralElement (implied)	PRP
Profile Registration	RegisteredProfile (central + scoping element)	CIM_RegisteredProfile	ScopingElement (implied)	SelfPRP
			CentralElement (implied)	SelfPRP
	ElementConformsToProfile	CIM_ElementConformsToProfile		
	ScopingElement	CIM_ManagedElement		
	CentralElement	CIM_ManagedElement		
	ReferencedProfile	CIM_ReferencedProfile		
	ReferencedRegisteredProfile	CIM_RegisteredProfile	RegisteredProfile	RefPRP

484 Table 15 lists the parts of the overall implementation that corresponds to the four profiles in the scenario, along with their profile implementation context and implemented advertisement methodology (in this example). The profile implementation context of each such part is defined by the profile reference in the referencing profile, and is stated as a path of named profile references relative to the top-level Example Base Server profile.

485

Table 15 – Profile related implementation parts in the SimpleStateDescription scenario

Profile Corresponding to the Implementation Part	Profile Implementation Context	Implemented Advertisement Methodology
Example Base Server	N/A (top-level)	central class methodology
Example Fan	SystemFan	central class methodology
Example Power Supply	SystemPowerSupply	scoping class methodology
Profile Registration	PRP	central class methodology
Profile Registration	SystemFan::PRP	central class methodology
Profile Registration	SystemPowerSupply::PRP	central class methodology
Profile Registration (1)	PRP::SelfPRP, SystemFan::PRP::SelfPRP, SystemPowerSupply::PRP:: SelfPRP	central class methodology

486 Note (1): This implementation uses an optimization for the implementation parts that correspond to this profile. The optimization uses one single RegisteredProfile instance to advertise conformance for all three parts; such optimizations are described in DSP1001.

487 Table 16 lists the implemented classes for this scenario.

488 **Table 16 – Implemented classes in the SimpleStateDescription scenario**

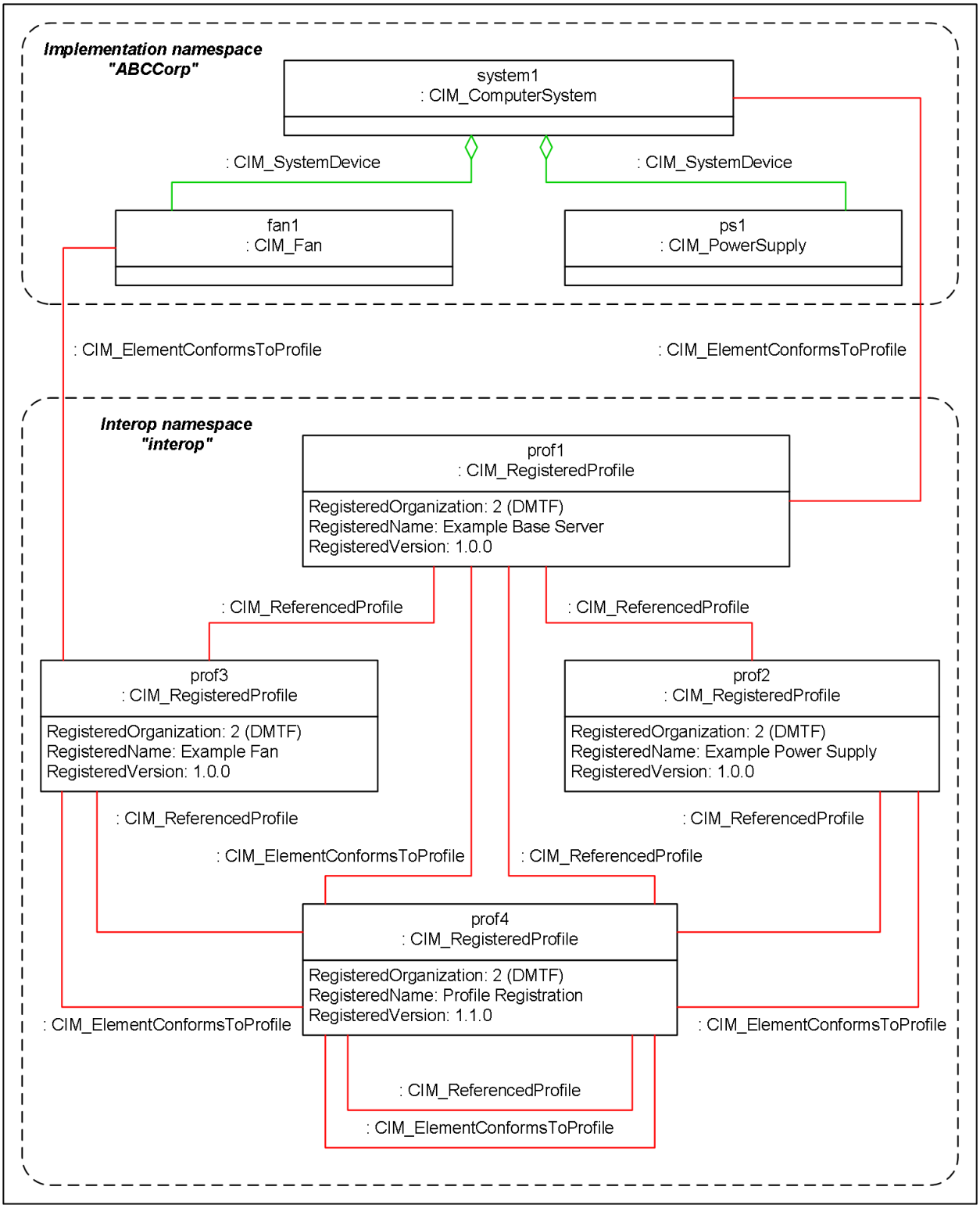
Implemented Class	Adaptation	Profile defining the Adaptation	Implementation Context for the Adaptation
CIM_ComputerSystem	ComputerSystem	Example Base Server	Example Base Server
	ScopingElement (implied)	Profile Registration	Example Base Server :: PRP
	CentralElement (implied)	Profile Registration	Example Base Server :: PRP
	System	Example Fan	Example Base Server :: SystemFan
	ScopingElement (implied)	Profile Registration	Example Base Server :: SystemFan :: PRP
	System	Example Power Supply	Example Base Server :: SystemPowerSupply
	ScopingElement (implied)	Profile Registration	Example Base Server :: SystemPowerSupply :: PRP
CIM_SystemDevice (for CIM_Fan)	SystemDevice	Example Fan	Example Base Server :: SystemFan
CIM_Fan	Fan	Example Fan	Example Base Server :: SystemFan
	CentralElement (implied)	Profile Registration	Example Base Server :: SystemFan :: PRP
CIM_SystemDevice (for CIM_PowerSupply)	SystemDevice	Example Power Supply	Example Base Server :: SystemPowerSupply
CIM_PowerSupply	PowerSupply	Example Power Supply	Example Base Server :: SystemPowerSupply
	CentralElement (implied)	Profile Registration	Example Base Server :: SystemPowerSupply :: PRP
CIM_ElementConformsToProfile (for central instances of Example Base Server profile)	ElementConformsToProfile	Profile Registration	Example Base Server :: PRP

Implemented Class	Adaptation	Profile defining the Adaptation	Implementation Context for the Adaptation
CIM_ElementConformsToProfile (for central instances of Example Fan profile)	ElementConformsToProfile	Profile Registration	Example Base Server :: SystemFan :: PRP
CIM_ElementConformsToProfile (for central instances of Profile Registration profile)	ElementConformsToProfile	Profile Registration	Example Base Server :: PRP :: SelfPRP, Example Base Server :: SystemFan :: PRP :: SelfPRP, Example Base Server :: SystemPowerSupply :: PRP :: SelfPRP
CIM_RegisteredProfile (for Example Base Server profile)	RegisteredProfile	Profile Registration	Example Base Server :: PRP
CIM_RegisteredProfile (for Example Fan profile)	ReferencedRegisteredProfile	Profile Registration	Example Base Server :: PRP
	RegisteredProfile	Profile Registration	Example Base Server :: SystemFan :: PRP
CIM_RegisteredProfile (for Example Power Supply profile)	ReferencedRegisteredProfile	Profile Registration	Example Base Server :: PRP
	RegisteredProfile	Profile Registration	Example Base Server :: SystemPowerSupply :: PRP
CIM_RegisteredProfile (for Profile Registration profile)	ReferencedRegisteredProfile	Profile Registration	Example Base Server :: PRP, Example Base Server :: SystemFan :: PRP, Example Base Server :: SystemPowerSupply :: PRP
	RegisteredProfile	Profile Registration	Example Base Server :: PRP :: SelfPRP, Example Base Server :: SystemFan :: PRP :: SelfPRP, Example Base Server :: SystemPowerSupply :: PRP :: SelfPRP
CIM_ReferencedProfile (for profiles referenced by Example Base Server profile)	ReferencedProfile	Profile Registration	Example Base Server :: PRP
CIM_ReferencedProfile (for profiles referenced by Example Fan profile)	ReferencedProfile	Profile Registration	Example Base Server :: SystemFan :: PRP
CIM_ReferencedProfile (for profiles referenced by Example Power Supply profile)	ReferencedProfile	Profile Registration	Example Base Server :: SystemPowerSupply :: PRP
CIM_ReferencedProfile (for profiles referenced by Profile Registration profile)	ReferencedProfile	Profile Registration	Example Base Server :: PRP, Example Base Server :: SystemFan :: PRP, Example Base Server :: SystemPowerSupply :: PRP

489 Note (1): This implementation is an optimization that merges three separate implementations into one implementation, as defined in DSP1001.

490 The object diagram in Figure 7 shows an example set of instances in this scenario. The implementation follows the recommendation to separate the implementation namespace from the Interop namespace.

491



492 **Figure 7 – Simple object diagram**

493

In this scenario, the `system1` instance representing a managed system, the `fan1` instance representing a fan in that system, and the `ps1` instance representing a power supply in that system are all exposed in the implementation namespace "ABCCorp".

494 The Interop namespace contains four instances of CIM_RegisteredProfile that advertise conformance to the Example Base Server, Example Fan, and Example Power Supply profiles, and to the Profile Registration profile (that is, this profile).

495 Profile conformance for the `ps1` instance is determined through the scoping class methodology because that instance is not referenced by any CIM_ElementConformsToProfile instances.

496 Profile conformance for the `fan1`, `system1` and the four CIM_RegisteredProfile instances is determined through the central class methodology because these instances are referenced by the ManagedElement end of a CIM_ElementConformsToProfile association instance.

497 Because some of the CIM_ElementConformsToProfile instances cross namespaces, the instances of these associations exist in both namespaces. The associated instances exist in only one of the namespaces. For example, the CIM_ElementConformsToProfile instance between `system1` and `prof1` has an instance in each of the two namespaces. In the instance in the implementation namespace, ManagedElement is a reference to the `system1` instance in the same namespace, and ConformantStandard is a cross-namespace reference to the `prof1` instance in the Interop namespace. In the instance in the Interop namespace, ConformantStandard is a reference to the `prof1` instance in the same namespace, and ManagedElement is a cross-namespace reference to the `system1` instance in the implementation namespace. See 6.4.4 for more information about cross-namespace associations.

498 The scenario defined in this state description is used by some of the following use cases.

499

8.2 Use case: RetrieveProfileInformationForComputerSystem

500 For the scenario defined in the SimpleStateDescription state description, this use case describes how a CIM client can retrieve profile information for an instance of CIM_ComputerSystem. In that scenario, the Example Base Server profile (defining the adaptation for the CIM_ComputerSystem class) is an autonomous profile.

501 This use case has the following preconditions:

503 • The instance path of a CIM_ComputerSystem instance (in the implementation namespace) is known.

505 • It is known that the Example Base Server profile is an autonomous profile and thus the implementation will always support the central class methodology.

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

508 1. Invoke the Associators operation on that CIM_ComputerSystem instance, filtering on the CIM_ElementConformsToProfile association class. The resulting CIM_RegisteredProfile instances represent all profiles to which that CIM_ComputerSystem instance conforms.

510 2. Iterate through the retrieved CIM_RegisteredProfile instances and inspect their RegisteredOrganization, RegisteredName and RegisteredVersion property values, which identify the profiles to which the CIM_ComputerSystem instance conforms.

511

8.3 Use case: RetrieveProfileVersionForFan

512 For the scenario defined in the SimpleStateDescription state description, this use case describes how a CIM client can retrieve the version of the Example Fan profile to which an instance of CIM_Fan conforms. In that scenario, the Example Fan profile (defining the adaptation for the CIM_Fan class) is a component profile and has been implemented using the central class methodology.

513

This use case has the following preconditions:

- The instance path of a CIM_Fan instance (in the implementation namespace) is known.
- It is known that the Example Fan profile is a component profile and that it has been implemented using the central class methodology.

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

1. Invoke the Associators operation on the given CIM_Fan instance, filtering on the CIM_ElementConformsToProfile association. This will retrieve all CIM_RegisteredProfile instances representing profiles to which that CIM_Fan instance conforms. In this scenario, only one CIM_RegisteredProfile instance representing the Example Fan profile will be returned.
2. The value of its RegisteredVersion property indicates the version of the Example Fan profile to which the given CIM_Fan instance conforms.

8.4 Use case: RetrieveProfileVersionForPowerSupply

For the scenario defined in the SimpleStateDescription state description, this use case describes how a CIM client can retrieve the version of the Example Power Supply profile to which an instance of the CIM_PowerSupply class conforms. In that scenario, the Example Power Supply profile (defining the adaptation for the CIM_PowerSupply class) is a component profile and has been implemented without implementing the central class methodology. As a result, the scoping class methodology is used.

This use case has the following preconditions:

- The instance path of a CIM_PowerSupply instance (in the implementation namespace) is known.
- It is known that the Example Power Supply profile is a component profile and that it has been implemented without implementing the central class methodology.

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

1. Navigate the scoping path defined in the Example Power Supply profile, from the central instance to the scoping instance, as follows:
    - Invoke the Associators operation on that CIM_PowerSupply instance, filtering on the CIM_SystemDevice association class. This will retrieve the (one) CIM_ComputerSystem instance that is the scoping instance of the CIM_PowerSupply instance.
2. Invoke the Associators operation on that CIM_ComputerSystem instance, filtering on the CIM_ElementConformsToProfile association. This will retrieve all CIM_RegisteredProfile instances representing profiles to which that CIM_ComputerSystem instance conforms. In this scenario, only one instance representing the Example Base Server profile will be returned.
3. Invoke the Associators operation on the returned CIM_RegisteredProfile instance representing the Example Base Server profile, filtering on the CIM_ReferencedProfile association class. This will retrieve all CIM_RegisteredProfile instances representing profiles referenced by the Example Base Server profile. In this scenario, three instances will be returned, representing the Example Power Supply, Example Fan, and Profile Registration profiles.
4. Iterate through these retrieved CIM_RegisteredProfile instances and select the Example Power Supply profile based on the values of its RegisteredOrganization and RegisteredName properties. The value of its RegisteredVersion property indicates the version of the Example Power supply profile to which the CIM_PowerSupply instance conforms.

8.5 Use case: AlgorithmForRetrievingProfileInformation

542 For the general case, this use case describes the algorithm for a CIM client to determine to which profiles a central instance of a given profile conforms, when the advertisement methodology implemented for that profile and for its scoping profiles is not known upfront.

543 This use case has the following preconditions:

545 • The instance path of a central instance of a given profile is known.

547 • The profile reference and scoping hierarchies between the given profile and its top-level autonomous profile is known, including the scoping path of each of those profiles.

548 Note that component profiles may define scoping elements that are not the central elements of their referencing profiles. For example, in the SimpleStateDescription scenario, the Example Fan profile could reference an additional Example Sensors profile that defines a scoping adaptation named System, that matches the ComputerSystem adaptation of the Example Base Server profile.

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

551 1. Invoke the Associators operation on the central instance, filtering on the CIM_ElementConformsToProfile association class.

553 2. If this operation returns one or more CIM_RegisteredProfile instances, the central class methodology has been implemented for the profile, and each (typically one) returned instance represents a profile to which the central instance advertises conformance (see the limitations described in 6.3.1).

554 The RegisteredOrganization, RegisteredName, and RegisteredVersion properties of the returned instances identify these profiles.

556 3. If this operation returns no CIM_RegisteredProfile instances, the central class methodology has not been implemented for the profile, and the scoping class methodology needs to be used. In that case, follow these steps:

558 • Starting with the central instance, invoke the Associators operation for each segment of the scoping path defined in the profile, filtering on the association classes and result classes, in order to navigate to its scoping instance.

560 • Invoke the Associators operation on that scoping instance, filtering on the CIM_ElementConformsToProfile association class. This returns the CIM_RegisteredProfile instances representing the profiles to which the scoping instance advertises conformance.

562 • If this operation returns one or more CIM_RegisteredProfile instances, the scoping profiles have been implemented using the central class methodology, and each (typically one) returned instance represents a profile to which the scoping instance advertises conformance.

563 Go to step 4.

565 • If this operation returns no CIM_RegisteredProfile instances, the scoping profiles also have been implemented using the scoping class methodology, and step 3 needs to be recursively repeated until a scoping instance is reached that returns such instances. After that is reached, each (typically one) returned instance represents a profile to which the scoping instance advertises conformance.

566 Go to step 4.

567

568 4. At this point, at least one CIM_RegisteredProfile instances representing profiles to which the top-most scoping instances advertise conformance.

569 Select the profile of those top-most profiles that directly or indirectly references the profile in which you are interested.

571 5. Invoke the Associators operation on the CIM_RegisteredProfile instance representing the selected top-most profile, filtering on the CIM_ReferencedProfile association class, and repeat that operation recursively on its result, such that you traverse as many profile levels down as you had to traverse profile levels up to the top-most profile in step 3. At each level, if more than one instance is returned, select the profile that directly or indirectly references the profile in question.

572 The CIM_RegisteredProfile instances resulting from the last such traversal represent the profiles to which the original central instance advertises conformance.

573 The RegisteredOrganization, RegisteredName, and RegisteredVersion properties of the returned instances identify these profiles.

574

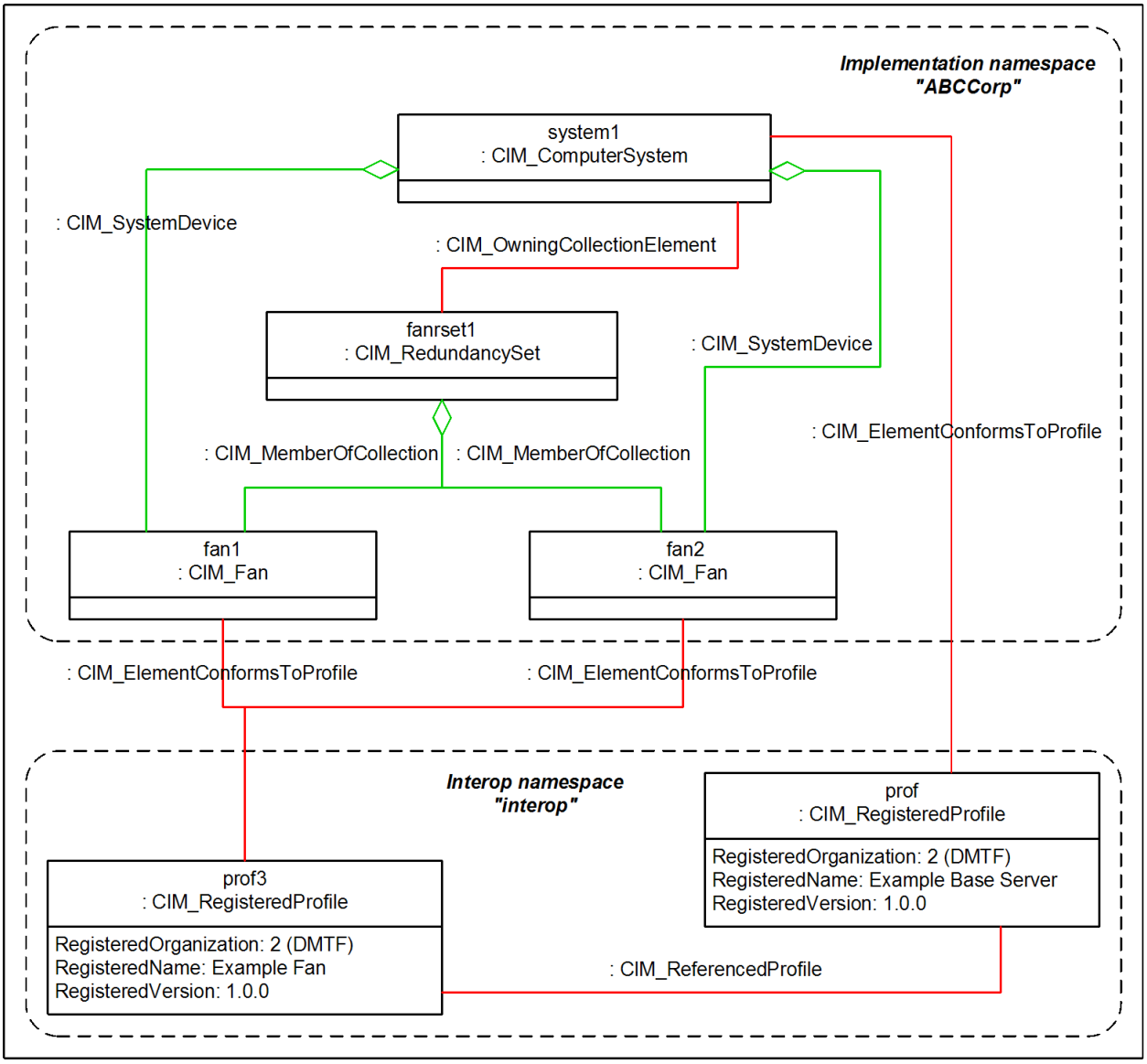
8.6 Use case: DetermineConformingInstances

575 Figure 8 is an object diagram for this use case and illustrates an implementation that conforms to the Example Fan profile described in the SimpleStateDescription scenario. The diagram shows some additional class adaptations defined in the Example Fan profile (compared to that scenario); schema classes are stated in the object diagram only for these additional adaptations. The central instances of the Example Fan profile are the two CIM_Fan instances, `fan1` and `fan2`.

576 The instances of adaptations defined in a profile form a graph, where those instances can be reached by association traversal from the central instances of that profile. Knowing the structure of this graph for the Example Fan profile, a CIM client can navigate to all these instances starting from the central instances of that profile, and can conclude from the existence of these instances that they conform to the Example Fan profile.

577 This use case determines all instances of ordinary adaptations conforming to the Example Fan profile, given the set of all central instances of that profile. Note that association instances conforming to the Example Fan profile are not determined in this use case; they could be determined by using the References operation.

578



579 **Figure 8 – Redundant fans object diagram**

580

This use case has the following preconditions:

582 • The instance paths of all central instances of the Example Fan profile are known.

584 • The navigation graph between instances of all adaptations defined in the Example Fan profile is known.

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

587 1. For each central instance and for each association adaptation defined in the Example Fan profile that starts at the Fan adaptation, invoke the Associators operation on that instance, filtering on the association class and result class of that association traversal. This will retrieve all conforming instances of ordinary classes one hop away from the central instance; in this case, the CIM_RedundancySet instance `fanrset1` and the CIM_RegisteredProfile instance `profile2`.

588

589 2. Repeat step 1 recursively for its resulting instances, until there are no more traversable adaptations defined in the Example Fan profile. This will retrieve the remaining set of conforming instances of ordinary classes; in this case, the CIM_ComputerSystem instance `system1`.

590 ## 8.7 Use case: AlgorithmForDeterminingAdvertisedProfiles

591 For the general case, this use case describes the algorithm for a CIM client to determine the set of profiles advertised by a WBEM server.

592 This use case has the following preconditions:

594 • The namespace path of the Interop namespace of the WBEM server is known.

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

597 1. Invoke the EnumerateInstances operation on the CIM_RegisteredProfile class in the Interop namespace.

598 This will retrieve the CIM_RegisteredProfile instances representing all profiles to which the WBEM server advertises conformance.

600 2. Iterate through these retrieved instances and inspect the values of their RegisteredOrganization, RegisteredName, and RegisteredVersion properties, which identify these profiles.

601 ## 8.8 Use case: AlgorithmForDeterminingTopLevelProfiles

602 For the general case, this use case describes the algorithm for a CIM client to determine the top-level profiles advertised by a WBEM server. Top-level profiles of an implementation are those that are not referenced by any other profiles to which the implementation conforms. This is accomplished by determining which instances of CIM_RegisteredProfile are not antecedents for any CIM_ReferencedProfile associations.

603 Typically, top-level profiles are autonomous profiles that represent the largest scoping of the CIM representation of the target system and that reference component profiles. Note that autonomous profiles may be referenced by other profiles.

604 This use case has the following preconditions:

606 • The namespace path of the Interop namespace of the WBEM server is known.

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

609 1. Invoke the EnumerateInstances operation on the CIM_RegisteredProfile class in the Interop namespace.

610 This will retrieve the CIM_RegisteredProfile instances representing all profiles to which the WBEM server advertises conformance.

612 2. Invoke the AssociatorNames operation on each of these CIM_RegisteredProfile instances, filtering on the CIM_ReferencedProfile association class and on source role Antecedent.

613 This will retrieve the instance paths of the CIM_RegisteredProfile instances representing all profiles to which the WBEM server advertises conformance and that are referenced by other such profiles.

615 3. Reduce the set of all profiles (retrieved in step 1) by the set of referenced profiles (retrieved in step 2), by means of comparing the values of their RegisteredOrganization, RegisteredName, and RegisteredVersion properties, which identify these profiles. This results in the set of all top-level profiles to which the WBEM server advertises conformance.

616

8.9 Use case: DetermineCentralInstancesForFan

617 For the scenario defined in the SimpleStateDescription state description, this use case describes how a CIM client can determine the central instances of the Example Fan profile. In that scenario, the Example Fan profile is a component profile and has implemented the central class methodology.

618 This use case has the following preconditions:

620 • The instance paths of any CIM_RegisteredProfile instances advertising conformance of the implementation to the Example Fan profile are known.

621 These instance paths can be determined as described in use case AlgorithmForDeterminingAdvertisedProfiles. Note that an implementation may expose more than one such instance.

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

624 1. For each CIM_RegisteredProfile instance for the Example Fan profile, invoke the Associators operation on that instance, filtering on the CIM_ElementConformsToProfile association class.

625 Because the Example Fan profile has implemented the central class methodology, the central instances of the Example Fan profile are returned.

626 If no instances are returned, the profile may not currently have any central instances. For example, the implementation may have chosen to represent pluggable fans as CIM_Fan instances only if they are plugged in, and the system may have no fans plugged in, currently. Note that older profiles require that an implementation exposes at least one central instance at any time.

628 2. Aggregate the central instances returned from all these invocations into one set.

629 This set is the set of central instances of the Example Fan profile, for this implementation.

630

8.10 Use case: DetermineCentralInstancesForPowerSupply

631 For the scenario defined in the SimpleStateDescription state description, this use case describes how a CIM client can determine the central instances of the Example Power Supply profile. In that scenario, the Example Power Supply profile is a component profile that does not have implemented the central class methodology. Therefore, this use case applies the scoping class methodology.

632 This use case has the following preconditions:

634 • The instance paths of any CIM_RegisteredProfile instances advertising conformance of the implementation to the Example Power Supply profile are known.

635 These instance paths can be determined as described in use case AlgorithmForDeterminingAdvertisedProfiles. Note that an implementation may expose more than one such instance.

637 • It is known that the scoping profile of the profile in question is an autonomous profile (in this scenario, the Example Base Server profile). Therefore, the central class methodology will be supported at the level of that scoping profile.

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

640 1. For each CIM_RegisteredProfile instance for the Example Power Supply profile, invoke the Associators operation on that instance, filtering on the CIM_ReferencedProfile association class and on source role Antecedent.

641

This will return CIM_RegisteredProfile instances for the Example Base Server profile. Aggregate the instances returned from all these invocations into one set, and reduce the set by eliminating any duplicate instances. Note that the resulting set may contain more than one instance.

643 2. For each instance in the resulting set, invoke the Associators operation on that instance, filtering on the CIM_ElementConformsToProfile association class.

644 Because the Example Base Server profile is an autonomous profile, the implementation will always use the central class methodology, and the central instances of the Example Base Server profile (that is, CIM_ComputerSystem instances) are returned.

645 If no instances are returned, the Example Base Server profile may not currently have any central instances. In this case, the Example Power Supply profile also has no central instances.

647 3. For each central instance of the Example Base Server profile, navigate across the scoping path of the Example Power Supply profile to its central instances by invoking the Associators operation on these instances, filtering on the CIM_SystemDevice association class, and on the CIM_PowerSupply result class.

648 Note that the filters used in this association traversal operation are tight enough to not return any undesired CIM_Fan instances.

650 4. Aggregate the CIM_PowerSupply instances returned from all these invocations into one set.

651 This set is the set of central instances of the Example Power Supply profile, for this implementation.

652

8.11 Use case: AlgorithmForDeterminingCentralInstancesOfProfile

653 This use case describes for the general case the algorithm for a CIM client to determine the central instances of a given profile that is advertised by a WBEM server, when the advertisement methodology implemented for that profile and for its scoping profiles is not known upfront.

654 This use case has the following preconditions:

656 • The namespace path of the Interop namespace of the WBEM server is known.

657 • The given profile is known by its registered name, organization, and version.

659 • The profile reference hierarchy between the given profile and its top-level autonomous profile is known, including the scoping path of each of those profiles.

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

662 1. Invoke the EnumerateInstances operation on the CIM_RegisteredProfile class in the Interop namespace.

663 This will retrieve the CIM_RegisteredProfile instances (and their instance paths) representing all profiles to which the WBEM server advertises conformance.

665 2. Out of the returned CIM_RegisteredProfile instances, determine the subset of instances where the values of their RegisteredOrganization, RegisteredName, and RegisteredVersion properties match the given profile.

666 If that subset contains more than one instance, repeat the following steps for each such instance. Note that there is no requirement that multiple implementations of the same profile in a WBEM server use the same CIM_RegisteredProfile instance for advertising conformance.

668 3. Navigate to the CIM_RegisteredProfile instance representing the next scoping profile that has implemented the central class methodology, by following these steps, starting from the CIM_RegisteredProfile instance:

- Invoke the Associators operation on the CIM_RegisteredProfile instance, filtering on association class CIM_ElementConformsToProfile.

  If one or more instances are returned, the profile has implemented the central class methodology (see the limitations described in 6.3.1); return from this recursive invocation of step 3.

  If no instances are returned, the profile did not implement the central class methodology. In that case, the scoping class methodology can be used. To do so, continue with the following steps.

- Invoke the Associators operation on the CIM_RegisteredProfile instance, filtering on the result role Dependent.

  This will return the CIM_RegisteredProfile instances representing the referencing profiles of the profile.

- Select the instance representing the scoping profile of the profile, utilizing knowledge about the profile reference tree.

- Recursively invoke step 3 for the CIM_RegisteredProfile instance representing the scoping profile of the profile.

2. Now that you have determined an instance of CIM_RegisteredProfile that represents the next scoping profile that uses the central class methodology . Invoke the Associators operation on that CIM_RegisteredProfile instance, filtering on the CIM_ElementConformsToProfile association class. This returns the central instances of that profile.

3. Based on knowledge about the scoping paths of each profile in the chain of referencing profiles whose CIM_RegisteredProfile instances were traversed in the previous steps, construct the effective scoping path between the originally given profile to the next scoping profile that uses the central class methodology.

Each of the central instances returned in step 4, is also a scoping instance in that effective scoping path. Navigate from each of these scoping instances across the effective scoping path to the central instances. The resulting instances are the central instances of the originally given profile.

8.1 Use case: AlgorithmForDeterminingCentral

For the general case, this use case describes the algorithm for a CIM client to determine whether a profile represented by a given CIM_RegisteredProfile instance has been implemented using the central class methodology.

This algorithm is based on whether CIM_ElementConformsToProfile associations are directly linked to the given instance of CIM_RegisteredProfile.

This use case has the following preconditions:

- The instance path of a CIM_RegisteredProfile instance (in the Interop namespace) is known.

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

1. Invoke the Associators operation on the given CIM_RegisteredProfile instance, filtering on the CIM_ElementConformsToProfile association class.

   If one or more instances are returned, the central class methodology is implemented for the registered profile (see the limitations described in 6.3.1).

   If no instances are returned, either the central class methodology has not been implemented, or it has been implemented but no central instance exists at this point.

Note, if the profile represented by the given CIM_RegisteredProfile instance is an autonomous profile, the central class methodology will always be available.

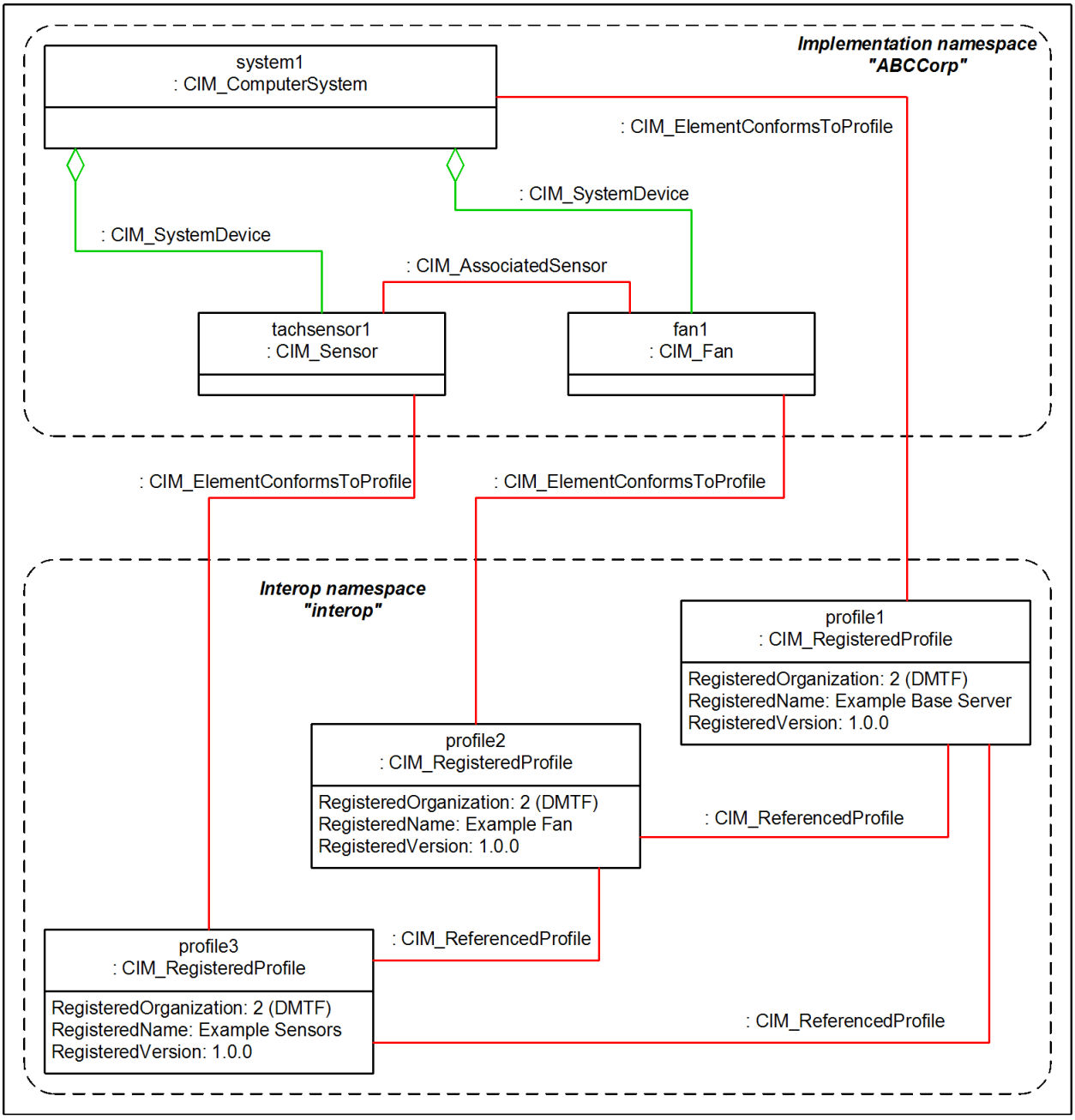
697

8.2 State description: PeerComponentProfileStateDescription

698 This scenario illustrates the relationship between CIM_RegisteredProfile instances for a component profile (Example Fan) that references another component profile (Example Sensors).

699 In this scenario, it is assumed that the Example Sensors profile has been implemented for speed sensors of the fans for which the Example Fan profile has been implemented. The Example Fan profile is the scoping profile for the Example Sensors profile, and the reference to the Example Sensors profile in the Example Fan profile is represented using CIM_ReferencedProfile instances between the respective CIM_RegisteredProfile instances.

700



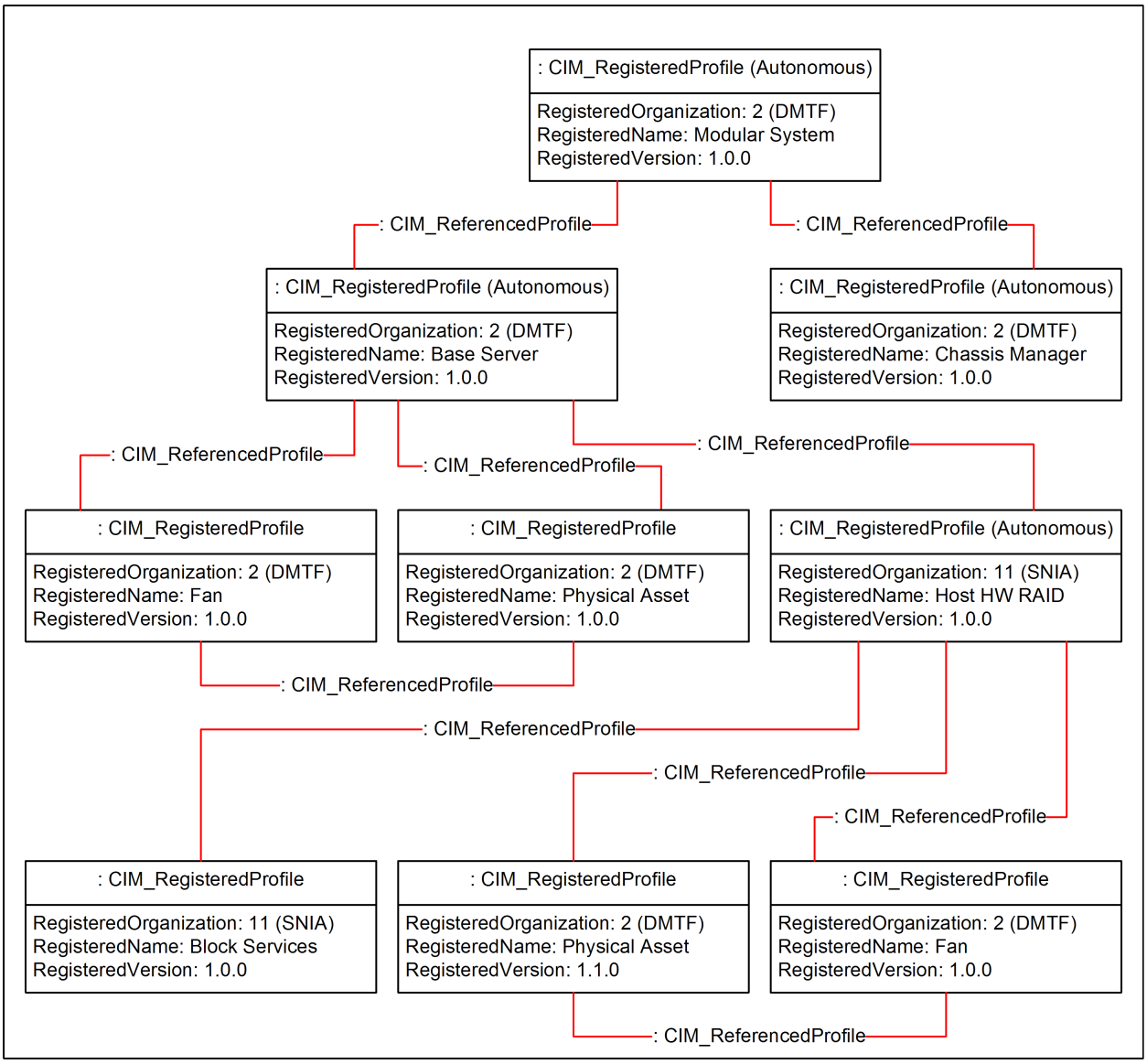
701 **Figure 9 – Referencing component profiles object diagram**

702

8.3 State description: ProfileComplianceHierarchyStateDescription

703 Figure 10 depicts the hierarchy of CIM_RegisteredProfile instances associated through CIM_ReferencedProfile instances that would represent a modular system with a chassis manager and an included blade server with RAID storage. This figure is provided as an example to illustrate the nature of the relationships among the various autonomous and component profiles. Also depicted are the relationships between component profiles.

704



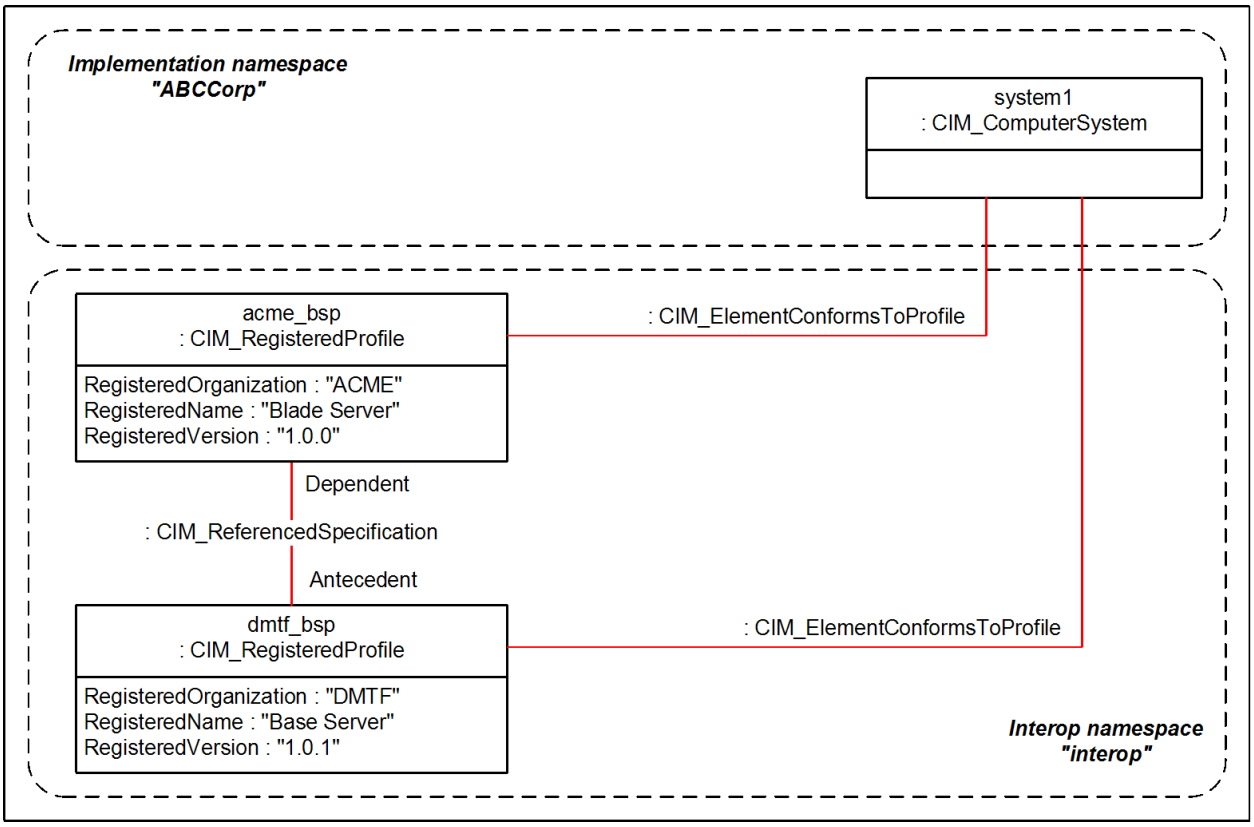
705 **Figure 10 – Profile compliance hierarchy object diagram**

706

8.4 State description: ProfileDerivationStateDescription

707 The object diagram in Figure 11 shows an implementation that conforms to a base profile and its derived profile.

708



709 **Figure 11 – Object diagram for profile derivation**

710

This diagram assumes a Blade Server profile defined by ACME that is derived from a Base Server profile defined by DMTF.

711 Conformance of the implementation to the ACME Blade Server profile is indicated by the `acme_bsp` instance, and conformance to the DMTF Base Server profile is indicated by the `dmtf_bsp` instance.

712 Because both of these profiles are autonomous profiles, the central and scoping path methodologies fall together causing the ElementConformsToProfile adaptation to be implemented for both profiles.

713 Because both profiles define CIM_ComputerSystem as their central element, each instance of CIM_ComputerSystem will be targeted by CIM_ElementConformsToProfile instances for both profiles.

714 Note that if conformance to a derived profile is advertised, it is not required that conformance to its base profile is also advertised. For example, the DMTF Base Server profile may in turn be derived from a DMTF Computer System profile which was chosen not to be advertised in this particular implementation.

715

ANNEX A
(informative)

Change log

Version	Date	Description
1.0.0	2007-06-25	

1.1.0	2014-05-22	Released as DMTF Standard with the following changes:
		• Converted to DMTF machine readable format. This included using new concepts from DSP1001 v1.0, such as class adaptations, features, constraints, generic operations and DMTF adaptation diagrams. The functionality of this profile in v1.1.0 is the same as in v1.0.0, it is just now described using these new concepts. Implementations that conformed to v1.0.0 of this profile, will also conform to v1.1.0 of this profile.
		• Added ability to represent the software identity of a profile implementation, as an optional feature.
		• Deprecated the use of leading slash (/) characters in namespace names. For producers of namespace names, tightened the permission to use a leading slash to become a recommendation against using a leading slash.
		• Deprecated the use of "root/interop" as a name for the Interop namespace.
		• Removed requirements on profile authoring, since these are now covered by DSP1001 v1.1. This caused the following v1.0 subclauses to be removed:
		  • "Central Class and Central Instance Identification"
		  • "Scoping Class and Scoping Instance Identification"
		  • "Association Traversal Path Existence"
		  • "Overlapping Profile Definitions"
		• Cleaned up terms and definitions. Deprecated the term "subject profile", replacing it with "registered profile".
		• Changes in use cases and state descriptions to better communicate the important scenarios.
		• Other small clarifications.
		• Changed version of CIM Schema to 2.39
		• Using the new generic operations names defined in DSP0223 1.0.2
		• Clarified confusing wording on the requirement to implement certain Interop namespace names (see 6.4.1).
		• Changed description of scoping methodology such that it is now described to be always available, and the central methodology is optionally in addition.
		• Simplified the definition of operation requirements for association traversal operations to define each operation only once per adaptation, that applies to all traversed associations starting on that adaptation.
		• Added requirement to implement the References and ReferenceNames association traversal operations.

Version	Date	Description
		• Using OCL conditions for a number of conditional properties. • Added support for determining the central instances using the GetCentralInstances() method. • Added overview section for profile relationships. • Fixed the requirement level of the ReferencedProfile and ReferencedRegisteredProfile adaptations to be Mandatory, consistent with v1.0. • Fixed the requirement levels of the version related properties of the SoftwareIdentity adaptation to be consistent with DSP1023 (Software Inventory Profile) • Changed the discovery definitions of the CentralClassMethodology and SoftwareIdentity features from text based to OCL based description. • Changed the requirement levels of the OtherRegisteredOrganization and AdvertiseTypeDescriptions properties of the RegisteredProfile adaptation from Mandatory and NullOk to Conditional with an OCLCondition that is based on the value of the companion property, to be more consistent with PUG 1.0 profiles. • Editorial improvements on the terms 'referenced profile' and 'referencing profile'.

Bibliography

745 DMTF DSP0206, *WBEM SLP Template 2.0*, http://www.dmtf.org/standards/published_documents/DSP0206_2.0.0.txt

746 DMTF DSP1054, *Indications Profile 1.2*, http://www.dmtf.org/standards/published_documents/DSP1054_1.2.pdf