



Document Number: DSP1033

Date: 2011-12-19

Version: 1.1.0a

Profile Registration Profile

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

This document expires on: **2012-05-31**.

Target version for DMTF Standard: **1.1.0**.

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

Document Type: Specification

Document Status: Work in Progress

Document Language: en-US

Copyright notice

Copyright © 2006-2011 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

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.

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patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a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

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>.

CONTENTS

Foreword

Introduction

1 Scope

2 Normative references

3 Terms and definitions

3.1 General

4 Symbols and abbreviated terms

5 Synopsis

6 Description

Collaboration structure diagram

Central and scoping class concept

General

Central class methodology

Scoping class methodology

WBEM server requirements on CIM namespaces

Interop namespace

Implementation namespaces

Relationship between Interop and implementation namespaces

Cross-namespace associations

7 Implementation

7.1 Features

7.1.1 Feature: CentralClassMethodology

7.1.2 Feature: ScopingClassMethodology

7.2 Adaptations

7.2.1 Conventions

7.2.2 Adaptation: RegisteredProfile: CIM_RegisteredProfile

7.2.3 Adaptation: ElementConformsToProfile: CIM_ElementConformsToProfile

7.2.4 Adaptation: ScopingElement: CIM_ManagedElement

7.2.5 Adaptation: CentralElement: CIM_ManagedElement

7.2.6 Adaptation: ReferencedProfile: CIM_ReferencedProfile

7.2.7 Adaptation: ReferencedRegisteredProfile: CIM_RegisteredProfile

8 Use cases and state descriptions

8.1 State description: SimpleStateDescription

8.2 Use case: RetrieveProfileInformationForComputerSystem

8.3 Use case: RetrieveProfileVersionForFan

8.4 Use case: RetrieveProfileVersionForPowerSupply

8.5 Use case: AlgorithmForRetrievingProfileInformation

8.6 Use case: DetermineConformingInstances

8.7 Use case: AlgorithmForDeterminingAdvertisedProfiles

8.8 Use case: AlgorithmForDeterminingTopLevelProfiles

- 8.9 Use case: AlgorithmForDeterminingCentralInstancesOfProfile
- 8.10 Use case: AlgorithmForDeterminingCentralOrScoping
- 8.11 State description: PeerComponentProfileStateDescription
- 8.12 State description: ProfileComplianceHierarchyStateDescription

Bibliography

Figures

- Figure 1 – DMTF collaboration structure diagram
- Figure 2 – Central class methodology example
- Figure 3 – Scoping class methodology example
- Figure 4 – Simple object diagram
- Figure 5 – Redundant fans object diagram
- Figure 6 – Peer component profiles object diagram
- Figure 7 – Profile compliance hierarchy object diagram

Tables

- Table 1 – Profile references
- Table 2 – Features
- Table 3 – Adaptations
- Table 4 – Use cases and state descriptions
- Table 5 – RegisteredProfile: Element requirements
- Table 6 – ElementConformsToProfile: Element requirements
- Table 7 – CentralElement: Element requirements
- Table 8 – ReferencedProfile: Element requirements
- Table 9 – ReferencedRegisteredProfile: Element requirements
- Table 10 – Profiles in the SimpleStateDescription scenario
- Table 11 – Adaptations in the SimpleStateDescription scenario
- Table 12 – Profile implementations in the SimpleStateDescription scenario
- Table 13 – Implemented adaptations in the SimpleStateDescription scenario
- Table 14 – Change log

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
- Christina Shaw, HP
- Paul von Behren, Symantec
- Mike Walker, IBM

Introduction

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.

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.

Document conventions

Typographical conventions

The following typographical conventions are used in this document:

- Document titles are marked in *italics*.
- Important terms that are used for the first time are marked in *italics*.
- Terms include a link to the term definition in the "Terms and definitions" clause, enabling easy navigation to the term definition.

OCL usage conventions

Constraints in this document are specified using OCL (see [OCL 2.0](#)).

OCL statements are in `monospaced font`.

Deprecated material

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.

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.

The following typographical convention indicates deprecated material:

DEPRECATED

Deprecated material appears here.

DEPRECATED

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

1 Scope

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.

2 Normative references

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.

DMTF DSP0004, *CIM Infrastructure Specification 2.6*,
http://www.dmtf.org/standards/published_documents/DSP0004_2.6.pdf

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

DMTF DSP1001, *Management Profile Specification Usage Guide 1.1*,
http://www.dmtf.org/standards/published_documents/DSP1001_1.1.pdf

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

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>

3 Terms and definitions

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

3.1 General

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. The terms "clause", "subclause", "paragraph", "annex" in this document are to be interpreted as described in [ISO/IEC Directives, Part2](#), Clause 5. The terms "normative" and "informative" in this document are to be interpreted as described in [ISO/IEC Directives, Part2](#), Clause 3. In this document, clauses, subclauses or annexes indicated with "(informative)" as well as notes and examples do not contain normative content. The terms defined in [DSP0004](#), [DSP0223](#), and [DSP1001](#) apply to this document.

The following additional terms are defined in this document.

3.2

autonomous profile

a profile that addresses an autonomous and self-contained management domain. For a complete definition, see [DSP1001](#).

[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 itself) and may scope component profiles.

See also term "component profile".

3.3

central class adaptation

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

3.4

central class methodology

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

3.5

central element

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

3.6

central instance

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

3.7

component profile

a profile that addresses a subset of a management domain. For a complete definition, see [DSP1001](#).

[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 in order to be implemented, and can be implemented only together with one if its scoping profiles. Component profiles may reference autonomous profiles and component profiles (including itself) and may scope other component profiles.

See also term "autonomous profile".

3.8

Interop namespace

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 subclause "Interop namespace". See also term "implementation namespace".

3.9

implementation namespace

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 subclause "Implementation namespaces". See also term "Interop namespace".

3.10**profile**

a management profile, as defined in [DSP1001](#).

3.11**profile conformance**

conformance of a profile implementation to its profile, such that it satisfies the rules for *full implementation conformance* defined in subclause 5.2.2 of [DSP1001](#).

3.12**referenced profile**

a profile that is referenced by a profile that lists it in its profile references table. For a complete definition, see subclause 7.9.1 of [DSP1001](#).

3.13**referencing profile**

a profile that references a profile by listing it in its profile references table. For a complete definition, see subclause 7.9.1 of [DSP1001](#).

3.14**registered profile**

a profile to which a profile implementation advertises conformance. Before version 1.1 of this profile, registered profiles were termed "subject profiles" (now deprecated).

3.15**scoping class adaptation**

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

3.16**scoping class methodology**

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

3.17**scoping element**

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

3.18**scoping instance**

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

3.19**scoping path**

an association traversal path between the central class adaptation and the scoping class adaptation. For a complete definition, see [DSP1001](#).

3.20**scoping profile**

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](#).

3.21

subject profile

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](#).

Use the term "registered profile" instead.

3.22

this profile

the profile defined in this profile specification (that is, the Profile Registration profile).

4 Symbols and abbreviated terms

The abbreviations defined in [DSP0004](#), [DSP0223](#), and [DSP1001](#) apply to this document.

This document does not define any additional abbreviations.

5 Synopsis

Profile name: Profile Registration

Version: 1.1.0

Organization: DMTF

Abstract indicator: False

Profile type: Autonomous

Schema: DMTF CIM 2.10

Central class adaptation: RegisteredProfile

Scoping class adaptation: RegisteredProfile

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.

For historical reasons, the scoping and central class adaptations of the Profile Registration profile are the same. Thus, it is an autonomous profile. Nonetheless, it cannot be implemented on its own, but only in context of its referencing profiles.

The following table identifies the profile references defined in this profile.

Table 1 – Profile references

Profile reference name	Profile name	Organization	Version	Relationship	Description
SelfPRP	Profile Registration	DMTF	1.0	Mandatory	Used to advertise profile conformance of an implementation of this profile itself.

The following table identifies the features defined in this profile.

Table 2 – Features

Feature	Requirement	Description
CentralClassMethodology	ConditionalExclusive	See subclause "Feature: CentralClassMethodology".
ScopingClassMethodology	ConditionalExclusive	See subclause "Feature: ScopingClassMethodology".

The following table identifies the class adaptations defined in this profile.

Table 3 – Adaptations

Adaptation	Elements	Requirement	Description
Instantiated, embedded and abstract adaptations			
RegisteredProfile	CIM_RegisteredProfile	Mandatory	See subclause "Adaptation: RegisteredProfile".
ElementConformsToProfile	CIM_ElementConformsToProfile	ConditionalExclusive	See subclause "Adaptation: ElementConformsToProfile".
ScopingElement	CIM_ManagedElement	See derived adaptations	See subclause "Adaptation: ScopingElement".
CentralElement	CIM_ManagedElement	See derived adaptations	See subclause "Adaptation: CentralElement".
ReferencedProfile	CIM_ReferencedProfile	ConditionalExclusive	See subclause "Adaptation: ReferencedProfile".
ReferencedRegisteredProfile	CIM_RegisteredProfile	See derived adaptations	See subclause "Adaptation: ReferencedRegisteredProfile".
Indications and exceptions			
This profile does not define any such adaptations.			

The following table identifies the use cases and state descriptions defined in this profile.

Table 4 – Use cases and state descriptions

Name	Description
State description SimpleStateDescription	See subclause "State description: SimpleStateDescription".
Use case RetrieveProfileInformationForComputerSystem	See subclause "Use case: RetrieveProfileInformationForComputerSystem".
Use case RetrieveProfileVersionForFan	See subclause "Use case: RetrieveProfileVersionForFan".
Use case RetrieveProfileVersionForPowerSupply	See subclause "Use case: RetrieveProfileVersionForPowerSupply".
Use case AlgorithmForRetrievingProfileInformation	See subclause "Use case: AlgorithmForRetrievingProfileInformation".
Use case DetermineConformingInstances	See subclause "Use case: DetermineConformingInstances".
Use case AlgorithmForDeterminingAdvertisedProfiles	See subclause "Use case: AlgorithmForDeterminingAdvertisedProfiles".
Use case AlgorithmForDeterminingTopLevelProfiles	See subclause "Use case: AlgorithmForDeterminingTopLevelProfiles".
Use case AlgorithmForDeterminingCentralInstancesOfProfile	See subclause "Use case: AlgorithmForDeterminingCentralInstancesOfProfile".
Use case AlgorithmForDeterminingCentralOrScoping	See subclause "Use case: AlgorithmForDeterminingCentralOrScoping".

Name	Description
State description PeerComponentProfileStateDescription	See subclause "State description: PeerComponentProfileStateDescription".
State description ProfileComplianceHierarchyStateDescription	See subclause "State description: ProfileComplianceHierarchyStateDescription".

6 Description

6.1 Collaboration structure diagram

DMTF collaboration structure diagrams show class adaptations (including association adaptations) and can express additional adaptation conformance requirements on an adaptation by means of *role bindings* (dotted lines) of *collaboration uses* (dotted ovals) that represent profile references and thus profile implementations. The names in a collaboration use oval that represents a profile reference are the profile reference name as defined in the profile reference table, and the profile name. The names on the role bindings of such collaboration uses are names of adaptations defined in the referenced profile represented by the collaboration use. These role bindings in the (informational) diagram are normatively defined as base adaptations of the adaptations targeted by the role bindings.

The following DMTF collaboration structure diagram shows all class adaptations defined in this profile, and all profiles referenced by this profile.

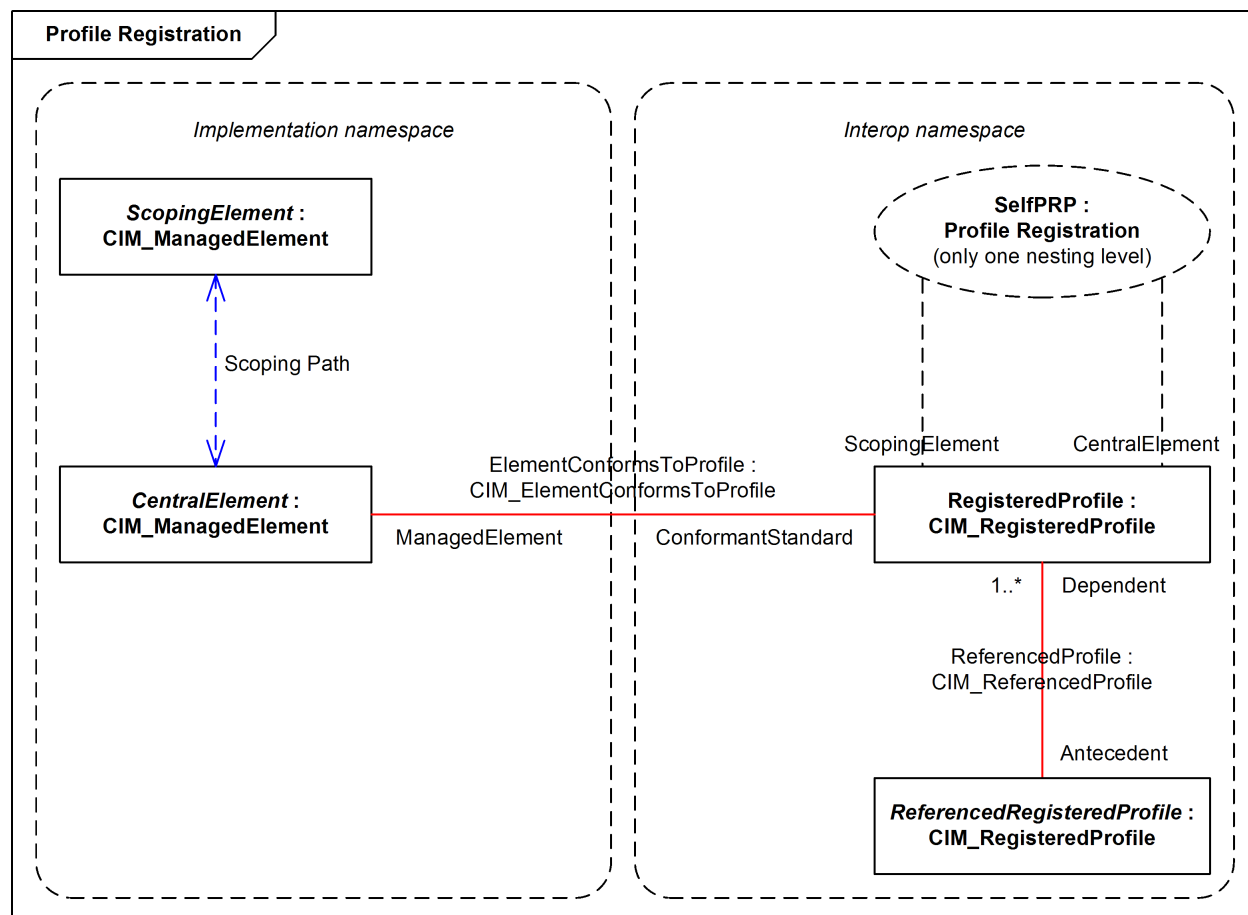


Figure 1 – DMTF collaboration structure diagram

Registered profiles are represented by instances of the RegisteredProfile adaptation in the Interop namespace.

As defined in subclause "WBEM server requirements on CIM namespaces", the roles of an Interop namespace and of an implementation namespace can be assumed by different namespaces or by the same namespace. The diagram in Figure 1 lists one namespace role for each dotted rounded rectangle. In WBEM servers that support implementation namespaces separate from their Interop namespace, instances of these adaptations would be exposed in their respective namespaces. In WBEM servers that support only one namespace, instances of these adaptations would all be exposed in that namespace.

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

The central and scoping elements of the registered profile are represented by instances of the CentralElement and ScopingElement adaptation, respectively, in the implementation namespace(s) of the registered profile.

If the registered profile implements the central class methodology, the ElementConformsToProfile association adaptation (between the RegisteredProfile and CentralElement adaptations) is implemented; otherwise, the registered profile implements the scoping class methodology and the ElementConformsToProfile association adaptation is not implemented. For a complete definition, see subclause "Central and scoping class concept".

If the registered profile references any profiles, the profiles to which the implementations of these referenced profiles conform are represented by instances of the ReferencedRegisteredProfile adaptation in the Interop namespace; these instances are associated via the ReferencedProfile association adaptation to the instances of the RegisteredProfile adaptation that represent the referencing profile. In addition, the ReferencedRegisteredProfile and ReferencedProfile adaptations may be implemented in the implementation namespaces of the registered profile.

The referenced profiles also advertise their profile conformance through an implementation of this profile. Because the referenced profiles of this profile are the registered profiles of those profile implementations, the ReferencedRegisteredProfile adaptation is based on the RegisteredProfile adaptation.

If the registered profile is a component profile, it has a scoping profile. Conformance of an implementation of the scoping profile to the scoping profile is again advertised by an implementation of this profile.

Because the central element of a scoping profile (of the registered profile) is the scoping element of the registered profile, the central element adaptation defined in the scoping profile is based on the scoping element adaptation defined in the registered profile.

The implementation of this profile itself is can also be advertised through a usage of this profile on itself; that is not shown in the diagram.

6.2 Central and scoping class concept

6.2.1 General

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. Since 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](#).

This profile establishes the concepts of a *central class adaptation* and a *scoping class adaptation*, in order for a client to be able to perform the following tasks:

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

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:

- Navigation between the RegisteredProfile instance representing a registered profile and its central instances is defined in this profile with *profile advertisement methodologies*; these do not require clients to have a priori knowledge about the particular profile.
- 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 mechanisms for that purpose.

Implementations that implement multiple profiles in a WBEM server (or even schema classes not covered by profiles) deserve particular attention by clients when navigating the network of instances of adaptations defined by a profile, since it is possible that instances that are not of adaptations defined by that profile are returned along with instances of adaptations defined by that profile. This often requires clients to have a priori knowledge about the way these multiple profiles have been combined in the implementation.

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.

In autonomous profiles, the central class adaptation and the scoping class adaptation are the same adaptation (see [DSP1001](#)).

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

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

See subclause "Central class methodology" for more information about the central class methodology.

- 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. As a result, the central instances of the registered profile are not directly associated through the ElementConformsToProfile adaptation to instances of the RegisteredProfile adaptation that represent the registered profile.

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.

This delegation may happen across multiple levels, 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.

See subclause "Scoping class methodology" for more information about the scoping class methodology.

The central class and scoping class methodologies are mutually exclusive for an implementation of a specific registered profile version; exactly one of them shall be implemented.

The decision on implementing central class or scoping class methodology should be left to the implementation; that is, profiles should not require one or the other profile advertisement methodology to be implemented.

In situations in which implementations have small footprint requirements and want to reduce the number of instances or in which the implementation is monolithic and the versions of implemented profiles is homogeneous, the implementation may use the scoping class methodology to reduce the number of necessary `ElementConformsToProfile` instances.

In situations in which multiple versions of the same profile are implemented, such as multi-vendor providers being integrated into a single WBEM server, the central class methodology is recommended to provide unambiguous relationships through `ElementConformsToProfile` instances between central instances and the `RegisteredProfile` instances representing the registered profiles with their versions.

For autonomous profiles, the scoping class methodology gets reduced to become the same as the central class methodology, because scoping element and central element are the same.

An implementation in which multiple versions of a registered profile are implemented may use different methodologies for each profile version, as long as the scoping class methodology is used for at most one of the profile versions implemented. The reason for this restriction is that with more than one use of the scoping class methodology, it is not possible to find out which subset of the central instances are related to which version of the registered profile. An example of this situation could be a system with two network interface cards, each from a different vendor that has delivered an implementation of the Ethernet Port Profile where the implementations are of different versions of the profile. This example 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 an implementation on the safe side in multi-vendor environments.

This profile defines no mechanisms for explicitly advertising which methodology has been used when implementing a registered profile, because the methodology can be ascertained by testing whether a central instance of the registered profile is referenced by an `ElementConformsToProfile` instance. Determining the methodology by testing whether the `RegisteredProfile` instance representing the registered profile is referenced by an `ElementConformsToProfile` instance only works when it is also ascertained that there is at least one central instance of the registered profile.

6.2.2 Central class methodology

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.

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.

Implementing this profile advertisement methodology is covered by the `CentralClassMethodology` feature.

Figure 2 is an object diagram (showing unnamed instances with their top-level class adaptation names) 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:

- 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
- from a RegisteredProfile instance that represents a registered profile to the central instances of that profile

In both cases, the traversal of the ElementConformsToProfile adaptation typically will be across namespaces; that is not represented in Figure 2 and is described in subclause "Cross-namespace associations".

In Figure 2, 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.

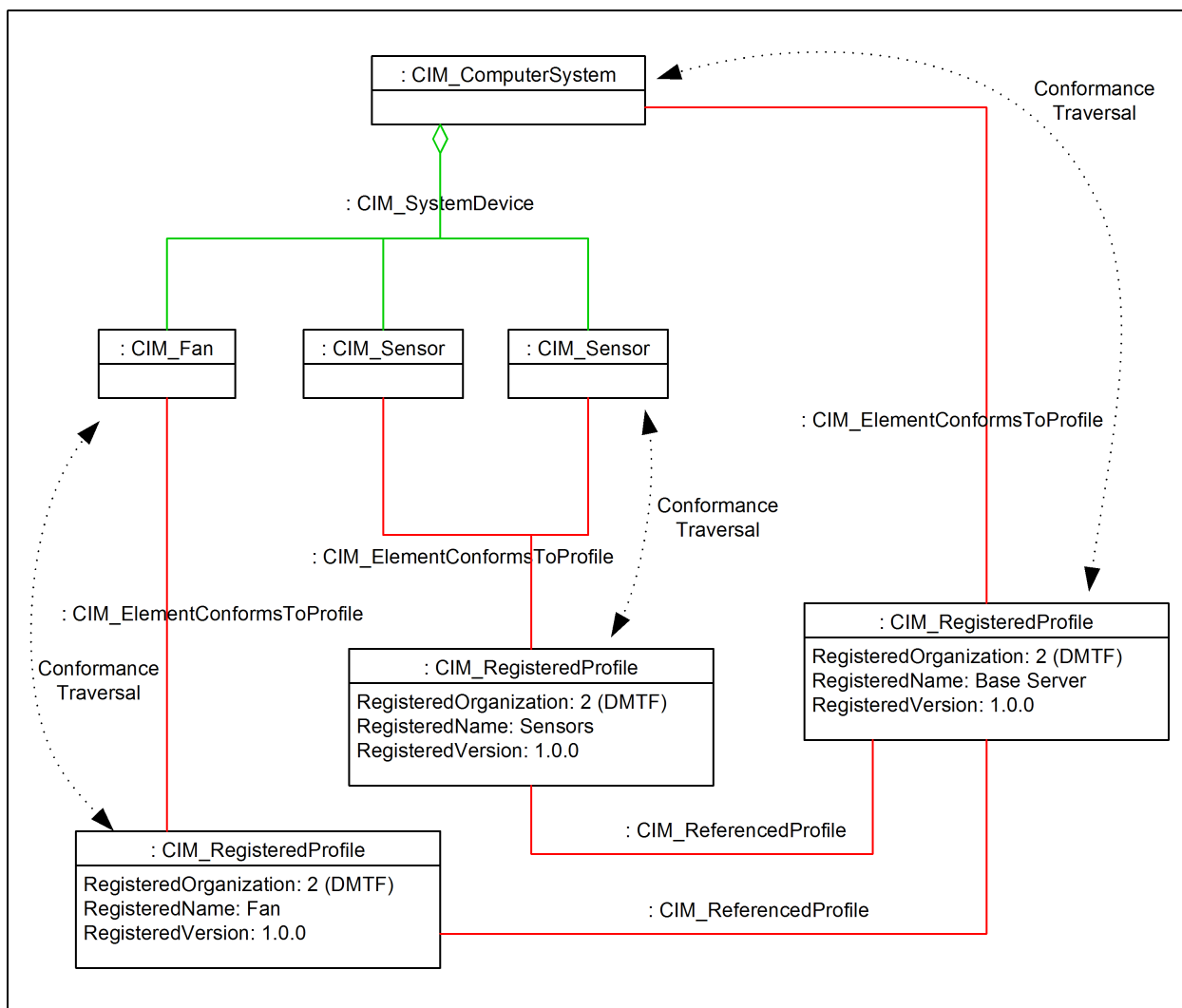


Figure 2 – Central class methodology example

6.2.3 Scoping class methodology

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 implementation and a RegisteredProfile instance that represents that registered profile, but instead by having that association adaptation at the next (directly or indirectly) scoping profile that implements the central class methodology for itself.

Implementing this profile advertisement methodology is covered by the ScopingClassMethodology feature.

Figure 3 is an object diagram (showing unnamed instances with their top-level class adaptation names) that provides an example of the scoping class methodology of advertising profile conformance with one level of scoping profiles.

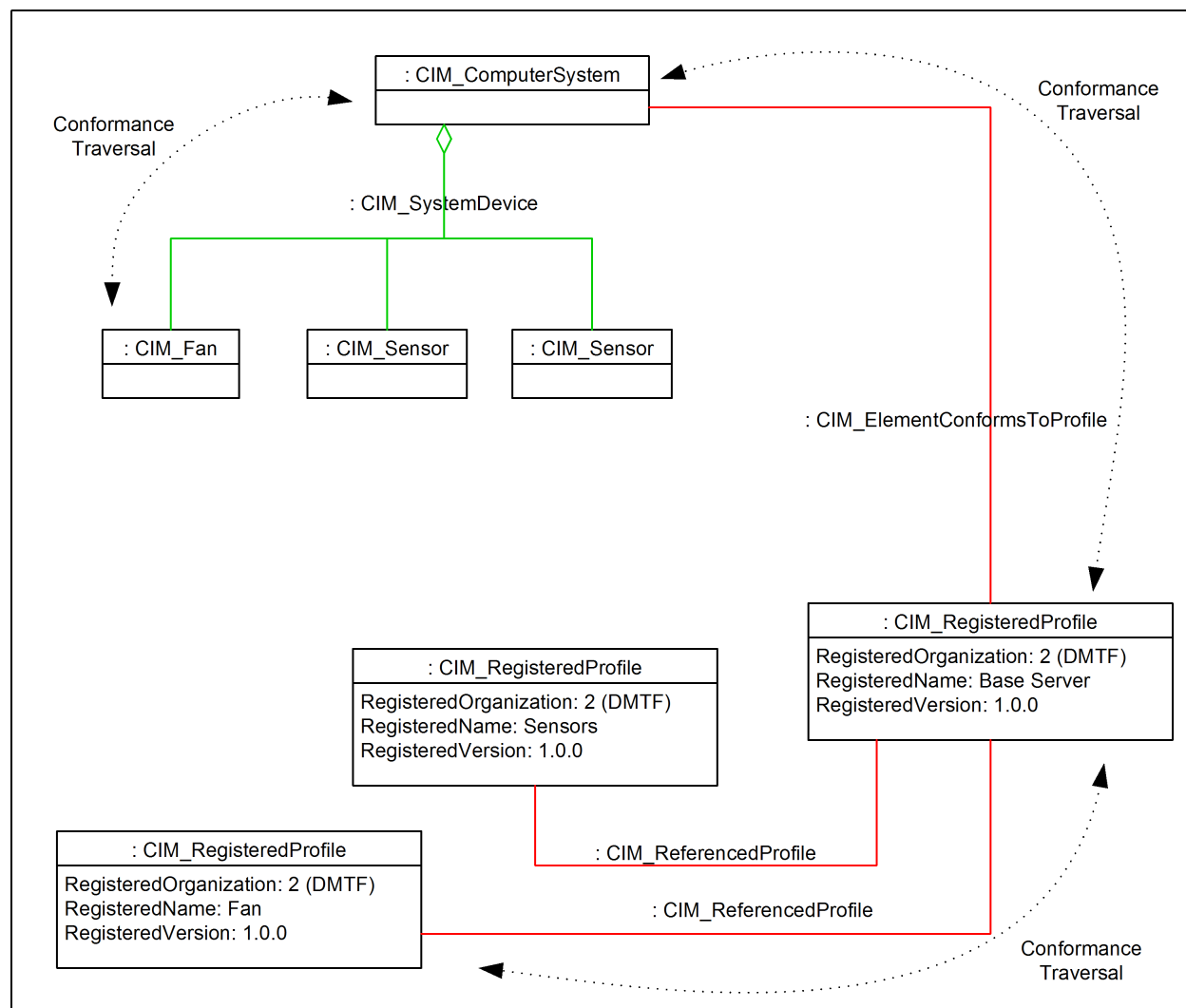


Figure 3 – Scoping class methodology example

In Figure 3, 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. Since the ComputerSystem instance is referenced by ElementConformsToProfile instances, the client knows that the corresponding profile has implemented 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 of the Fan instance is advertising conformance.

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.

The concept is in both cases that the client navigates up the scoping profile hierarchy to the level where a scoping profile implements the central class methodology (as indicated by the presence of instances of the ElementConformsToProfile association adaptation), 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.

In both cases, the traversal of the ElementConformsToProfile adaptation typically will be across namespaces; that is not represented in Figure 3 and is described in subclause "Cross-namespace associations".

In Figure 3, 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.

6.3 WBEM server requirements on CIM namespaces

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

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.

6.3.1 Interop namespace

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.

A WBEM server shall implement exactly one CIM namespace that assumes the role of an Interop namespace; that namespace is also called the Interop namespace.

A WBEM server shall expose its Interop namespace using the namespace name:

```
interop
```

DEPRECATED

A WBEM server may expose its Interop namespace using the following alternative namespace name, instead of using the "interop" namespace name:

```
root/interop
```

The use of this alternative namespace name is not preferred and has been deprecated in version 1.1 of this profile.

Note that clients need to be prepared to deal with any one of these two namespace names.

DEPRECATED

A WBEM server may expose its Interop namespace using additional implementation-defined namespace names that are not one of the namespace names described previously in this subclause. This accommodates WBEM server implementations that support namespace alias names. The client-visible

appearance of such a WBEM server is that it exposes multiple distinct Interop namespaces, each with a distinct set of CIM objects (where these sets are equal, except for different CIM object paths).

DEPRECATED

The use of leading slash (/) characters in Interop namespace names is deprecated.

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. Version 1.0 of this profile permitted a leading slash character in the name of the Interop namespace. [DSP0004](#) does not permit namespace names to begin with a slash. Therefore, version 1.1 of this profile has deprecated the use of leading slash characters in the name of the Interop namespace.

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

6.3.2 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.

6.3.3 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 subclause "Interop namespace".

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.

A profile implementation may span multiple namespaces, including multiple implementation namespaces.

6.3.4 Cross-namespace associations

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

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.

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.

7 Implementation

7.1 Features

7.1.1 Feature: CentralClassMethodology

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 subclause "Central class methodology".

The requirement level for this feature is conditional exclusive, with the following condition:

The following is NOT true:

- The ScopingClassMethodology feature is implemented.

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:

- At least one ElementConformsToProfile instance exists that references the RegisteredProfile instance representing the registered profile. This discovery mechanism only works if multiple implementations of the registered profile that advertise through the same RegisteredProfile instance implement the same methodology.

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

The following profile elements are conditional or conditional exclusive on the implementation of this feature:

- ScopingClassMethodology feature.
- ElementConformsToProfile adaptation.
- GetAssociatedInstancesWithPath() operation in RegisteredProfile adaptation.
- GetAssociatedInstancePaths() operation in RegisteredProfile adaptation.
- GetAssociatedInstancesWithPath() operation in CentralElement adaptation.
- GetAssociatedInstancePaths() operation in CentralElement adaptation.

7.1.2 Feature: ScopingClassMethodology

Implementing this feature for a registered profile provides support for advertising conformance of an implementation to that registered profile using the scoping class methodology. For details, see subclause "Scoping class methodology".

The requirement level for this feature is conditional exclusive, with the following condition:

The following is NOT true:

- The CentralClassMethodology feature is implemented.

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:

- No ElementConformsToProfile instance exists that references the RegisteredProfile instance representing the registered profile. This discovery mechanism only works if multiple implementations of the registered profile that advertise through the same RegisteredProfile instance implement the same methodology.

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

The following profile elements are conditional or conditional exclusive on the implementation of this feature:

- CentralClassMethodology feature.

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 the "Adaptations" clause.

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

This adaptation models registered profiles.

It is important to understand that representing the conformance of a profile implementation to a profile is not the same as representing the profile implementation itself. For example, multiple implementations of the same profile can exist that use the same instance of this adaptation to represent conformance to that profile. Alternatively, each of such multiple implementations of a profile can have its own instance of this adaptation to represent conformance to the profile, even when these profiles have the same version and these implementations use the same implementation namespace.

The implementation type of this adaptation is instantiated ordinary adaptation.

The requirement level for this adaptation is mandatory.

The following table identifies the element requirements for this adaptation.

Table 5 – RegisteredProfile: Element requirements

Element	Requirement	Description
Base adaptations		
SelfPRP::CentralElement	ConditionalExclusive	See SelfPRP::CentralElement. Condition: This base adaptation causes a recursive implementation of this profile; advertising that the outer implementation of this profile conforms to this profile. In order to avoid an endless recursion of implementations of this profile, the condition for this base adaptation is that it is true at the first level of that recursion, and false at any further levels.
SelfPRP::ScopingElement	ConditionalExclusive	See SelfPRP::ScopingElement. Condition: This base adaptation causes a recursive implementation of this profile; advertising that the outer implementation of this profile conforms to this profile. In order to avoid an endless recursion of implementations of this profile, the condition for this base adaptation is that it is true at the first level of that recursion, and false at any further levels.
Properties		
InstanceID	Mandatory	Key, see schema definition.
RegisteredOrganization	Mandatory	Required, see schema definition.
RegisteredName	Mandatory	Required, see subclause "Property: RegisteredName".
RegisteredVersion	Mandatory	Required, see schema definition.
AdvertiseTypes	Mandatory	Required, see schema definition.
OtherRegisteredOrganization	Conditional	See subclause "Property: OtherRegisteredOrganization".
AdvertiseTypeDescriptions	Conditional	See subclause "Property: AdvertiseTypeDescriptions".
Operations		
GetInstance()	Mandatory	See DSP0223 .
GetClassInstancesWithPath()	Mandatory	See DSP0223 .
GetClassInstancePaths()	Mandatory	See DSP0223 .
GetAssociatedInstancesWithPath() for ElementConformsToProfile	ConditionalExclusive	See subclause "Operation: GetAssociatedInstancesWithPath() for ElementConformsToProfile".
GetAssociatedInstancePaths() for ElementConformsToProfile	ConditionalExclusive	See subclause "Operation: GetAssociatedInstancePaths() for ElementConformsToProfile".

Element	Requirement	Description
GetAssociatedInstancesWithPath() for ReferencedProfile	ConditionalExclusive	See subclause "Operation: GetAssociatedInstancesWithPath() for ReferencedProfile".
GetAssociatedInstancePaths() for ReferencedProfile	ConditionalExclusive	See subclause "Operation: GetAssociatedInstancePaths() for ReferencedProfile".

7.2.2.2 Property: RegisteredName

The presentation requirement level for this property is mandatory.

The value shall be the name of the registered profile.

7.2.2.3 Property: OtherRegisteredOrganization

The presentation requirement level for this property is conditional, with the following condition:

The RegisteredOrganization property can potentially have a value of 1 (Other).

7.2.2.4 Property: AdvertiseTypeDescriptions

The presentation requirement level for this property is conditional, with the following condition:

The AdvertiseTypes property can potentially have a value of 1 (Other).

7.2.2.5 Operation: GetAssociatedInstancesWithPath() for ElementConformsToProfile

For general requirements on the implementation of this operation, see [DSP0223](#).

The requirement level for this operation is conditional exclusive, with the following condition:

The CentralClassMethodology feature is implemented.

This operation requirement applies when traversing the following association adaptations:

- ElementConformsToProfile

7.2.2.6 Operation: GetAssociatedInstancePaths() for ElementConformsToProfile

For general requirements on the implementation of this operation, see [DSP0223](#).

The requirement level for this operation is conditional exclusive, with the following condition:

The CentralClassMethodology feature is implemented.

This operation requirement applies when traversing the following association adaptations:

- ElementConformsToProfile

7.2.2.7 Operation: GetAssociatedInstancesWithPath() for ReferencedProfile

For general requirements on the implementation of this operation, see [DSP0223](#).

The requirement level for this operation is conditional exclusive, with the following condition:

This profile is implemented for a profile referenced by the registered profile.

This operation requirement applies when traversing the following association adaptations:

- ReferencedProfile

7.2.2.8 Operation: GetAssociatedInstancePaths() for ReferencedProfile

For general requirements on the implementation of this operation, see [DSP0223](#).

The requirement level for this operation is conditional exclusive, with the following condition:

This profile is implemented for a profile referenced by the registered profile.

This operation requirement applies when traversing the following association adaptations:

- ReferencedProfile

7.2.3 Adaptation: ElementConformsToProfile: CIM_ElementConformsToProfile

7.2.3.1 General

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

The implementation type of this adaptation is instantiated association adaptation.

The requirement level for this adaptation is conditional exclusive, with the following condition:

The CentralClassMethodology feature is implemented.

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 subclause "Central and scoping class concept".

The following table identifies the element requirements for this adaptation.

Table 6 – ElementConformsToProfile: Element requirements

Element	Requirement	Description
Properties		
ConformantStandard	Mandatory	Key, see subclause "Property: ConformantStandard".
ManagedElement	Mandatory	Key, see subclause "Property: ManagedElement".
Operations		
GetInstance()	Mandatory	See DSP0223 .

7.2.3.2 Property: ConformantStandard

The presentation requirement level for this property is mandatory.

The implementation shall satisfy the following constraints for this reference property:

- Referenced instances shall be of class adaptation RegisteredProfile.
- The multiplicity of [0 .. *] defined in the schema is not further constrained.

7.2.3.3 Property: ManagedElement

The presentation requirement level for this property is mandatory.

The implementation shall satisfy the following constraints for this reference property:

- Referenced instances shall be of class adaptation CentralElement.
- The multiplicity of [0 .. *] defined in the schema is not further constrained.

7.2.4 Adaptation: ScopingElement: CIM_ManagedElement

This adaptation models scoping elements of registered profiles.

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.

The implementation type of this adaptation is abstract ordinary adaptation.

The requirement level for this abstract adaptation is left to be defined in its derived adaptations.

7.2.5 Adaptation: CentralElement: CIM_ManagedElement

7.2.5.1 General

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.

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.

The implementation type of this adaptation is abstract ordinary adaptation.

The requirement level for this abstract adaptation is left to be defined in its derived adaptations.

The following table identifies the element requirements for this adaptation.

Table 7 – CentralElement: Element requirements

Element	Requirement	Description
Operations		
GetAssociatedInstancesWithPath() for ElementConformsToProfile	ConditionalExclusive	See subclause "Operation: GetAssociatedInstancesWithPath() for ElementConformsToProfile".
GetAssociatedInstancePaths() for ElementConformsToProfile	ConditionalExclusive	See subclause "Operation: GetAssociatedInstancePaths() for ElementConformsToProfile".

7.2.5.2 Operation: GetAssociatedInstancesWithPath() for ElementConformsToProfile

For general requirements on the implementation of this operation, see [DSP0223](#).

The requirement level for this operation is conditional exclusive, with the following condition:

The CentralClassMethodology feature is implemented.

This operation requirement applies when traversing the following association adaptations:

- ElementConformsToProfile

7.2.5.3 Operation: GetAssociatedInstancePaths() for ElementConformsToProfile

For general requirements on the implementation of this operation, see [DSP0223](#).

The requirement level for this operation is conditional exclusive, with the following condition:

The CentralClassMethodology feature is implemented.

This operation requirement applies when traversing the following association adaptations:

- ElementConformsToProfile

7.2.6 Adaptation: ReferencedProfile: CIM_ReferencedProfile

7.2.6.1 General

This adaptation models the relationship between registered profiles and the profiles they reference.

The implementation type of this adaptation is instantiated association adaptation.

The requirement level for this adaptation is conditional exclusive, with the following condition:

This profile is implemented for a profile referenced by the registered profile.

The following table identifies the element requirements for this adaptation.

Table 8 – ReferencedProfile: Element requirements

Element	Requirement	Description
Properties		
Antecedent	Mandatory	Key, see subclause "Property: Antecedent".
Dependent	Mandatory	Key, see subclause "Property: Dependent".
Operations		
GetInstance()	Mandatory	See DSP0223 .

7.2.6.2 Property: Antecedent

The presentation requirement level for this property is mandatory.

The implementation shall satisfy the following constraints for this reference property:

- Referenced instances shall be of class adaptation ReferencedRegisteredProfile.
- The multiplicity of [0 .. *] defined in the schema is not further constrained.

7.2.6.3 Property: Dependent

The presentation requirement level for this property is mandatory.

The implementation shall satisfy the following constraints for this reference property:

- Referenced instances shall be of class adaptation RegisteredProfile.
- The multiplicity of [0 .. *] defined in the schema is not further constrained.

7.2.7 Adaptation: ReferencedRegisteredProfile: CIM_RegisteredProfile

7.2.7.1 General

This adaptation models referenced profiles; that is, profiles that are referenced by the registered profile for which the RegisteredProfile adaptation is implemented. The type of profile relationship can be any; that is, usage or derivation. This adaptation provides the ability to traverse the ReferencedProfile association between the RegisteredProfile instances representing the referencing profile and the referenced profile.

This adaptation shall be (implicitly) applied as a base adaptation to the RegisteredProfile adaptation when implemented in the profile implementation context of the referenced profile. In other words, that adaptation is considered a derived adaptation of this adaptation in a different profile implementation context. For example, if an Example Fan profile references an Example Sensors profile, and both reference this profile, the implementation of the RegisteredProfile adaptation in context of the Example Sensors profile's use of this profile will get the ReferencedRegisteredProfile adaptation in context of the Example Fan profile's use of this profile applied.

The implementation type of this adaptation is abstract ordinary adaptation.

The requirement level for this abstract adaptation is left to be defined in its derived adaptations.

The following table identifies the element requirements for this adaptation.

Table 9 – ReferencedRegisteredProfile: Element requirements

Element	Requirement	Description
Operations		
GetAssociatedInstancesWithPath() for ReferencedProfile	ConditionalExclusive	See subclause "Operation: GetAssociatedInstancesWithPath() for ReferencedProfile".
GetAssociatedInstancePaths() for ReferencedProfile	ConditionalExclusive	See subclause "Operation: GetAssociatedInstancePaths() for ReferencedProfile".

7.2.7.2 Operation: GetAssociatedInstancesWithPath() for ReferencedProfile

For general requirements on the implementation of this operation, see [DSP0223](#).

The requirement level for this operation is conditional exclusive, with the following condition:

This profile is implemented for a profile referenced by the registered profile.

This operation requirement applies when traversing the following association adaptations:

- ReferencedProfile

7.2.7.3 Operation: GetAssociatedInstancePaths() for ReferencedProfile

For general requirements on the implementation of this operation, see [DSP0223](#).

The requirement level for this operation is conditional exclusive, with the following condition:

This profile is implemented for a profile referenced by the registered profile.

This operation requirement applies when traversing the following association adaptations:

- ReferencedProfile

8 Use cases and state descriptions

8.1 State description: SimpleStateDescription

This state description describes a simple scenario in which three example profiles have been implemented. Each of them advertises conformance through an implementation of this profile (i.e. the Profile Registration profile). Each implementation of this profile in turn advertises conformance to this profile itself.

The following table lists these three example profiles and for clarity also this profile, and the profiles they reference:

Table 10 – 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

The following table lists the class adaptations defined in the three example profiles and in this profile, to the extent they are relevant for this scenario.

Table 11 – 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
			ReferencedRegisteredProfile (implied)	Profile Registration profile implementation in context of referenced profile
	ElementConformsToProfile	CIM_ElementConformsToProfile		
	ScopingElement	CIM_ManagedElement		
	CentralElement	CIM_ManagedElement		
	ReferencedProfile	CIM_ReferencedProfile		
	ReferencedRegisteredProfile	CIM_RegisteredProfile		

The following table lists the implementations of these profiles in the scenario, along with their profile implementation context and implemented advertisement methodology. The profile implementation context of each implemented profile 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 implementation of the Example Base Server profile.

Table 12 – Profile implementations in the SimpleStateDescription scenario

Implemented Profile	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

Note (1): This implementation of this profile is an optimization that merges three separate implementations into one implementation, as defined in [DSP1001](#).

The following table lists the implemented class adaptations that are used in the object diagram.

Table 13 – Implemented adaptations in the SimpleStateDescription scenario

Implemented Adaptation	Profile Implementation Context	Base Adaptation (effective)	Profile Implementation Context (of base adaptation)
ComputerSystem	N/A (top-level)	ScopingElement (implied)	PRP
		CentralElement (implied)	PRP
		System	SystemFan
		ScopingElement (implied)	SystemFan::PRP
		System	SystemPowerSupply
		ScopingElement (implied)	SystemPowerSupply::PRP
SystemDevice	SystemFan		
SystemDevice	SystemPowerSupply		
Fan	SystemFan	CentralElement (implied)	SystemFan::PRP
PowerSupply	SystemPowerSupply	CentralElement (implied)	SystemPowerSupply::PRP
ElementConformsToProfile	PRP		
ElementConformsToProfile	SystemFan::PRP		
ElementConformsToProfile	PRP::SelfPRP		
ElementConformsToProfile	SystemFan::PRP::SelfPRP		
ElementConformsToProfile	SystemPowerSupply::PRP::SelfPRP		
RegisteredProfile	PRP		

Implemented Adaptation	Profile Implementation Context	Base Adaptation (effective)	Profile Implementation Context (of base adaptation)
RegisteredProfile	SystemFan::PRP	ReferencedRegisteredProfile (implied)	PRP
RegisteredProfile	SystemPowerSupply::PRP	ReferencedRegisteredProfile (implied)	PRP
RegisteredProfile (1)	PRP::SelfPRP SystemFan::PRP::SelfPRP SystemPowerSupply::PRP::SelfPRP	ReferencedRegisteredProfile (implied)	PRP SystemFan::PRP SystemPowerSupply::PRP
ReferencedProfile	PRP		
ReferencedProfile	SystemFan::PRP		
ReferencedProfile	SystemPowerSupply::PRP		

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

The following object diagram shows an example set of instances in this scenario. The implementation follows the recommendation to separate the implementation namespace from the Interop namespace, and it implements the requirement to advertise conformance to this profile through itself.

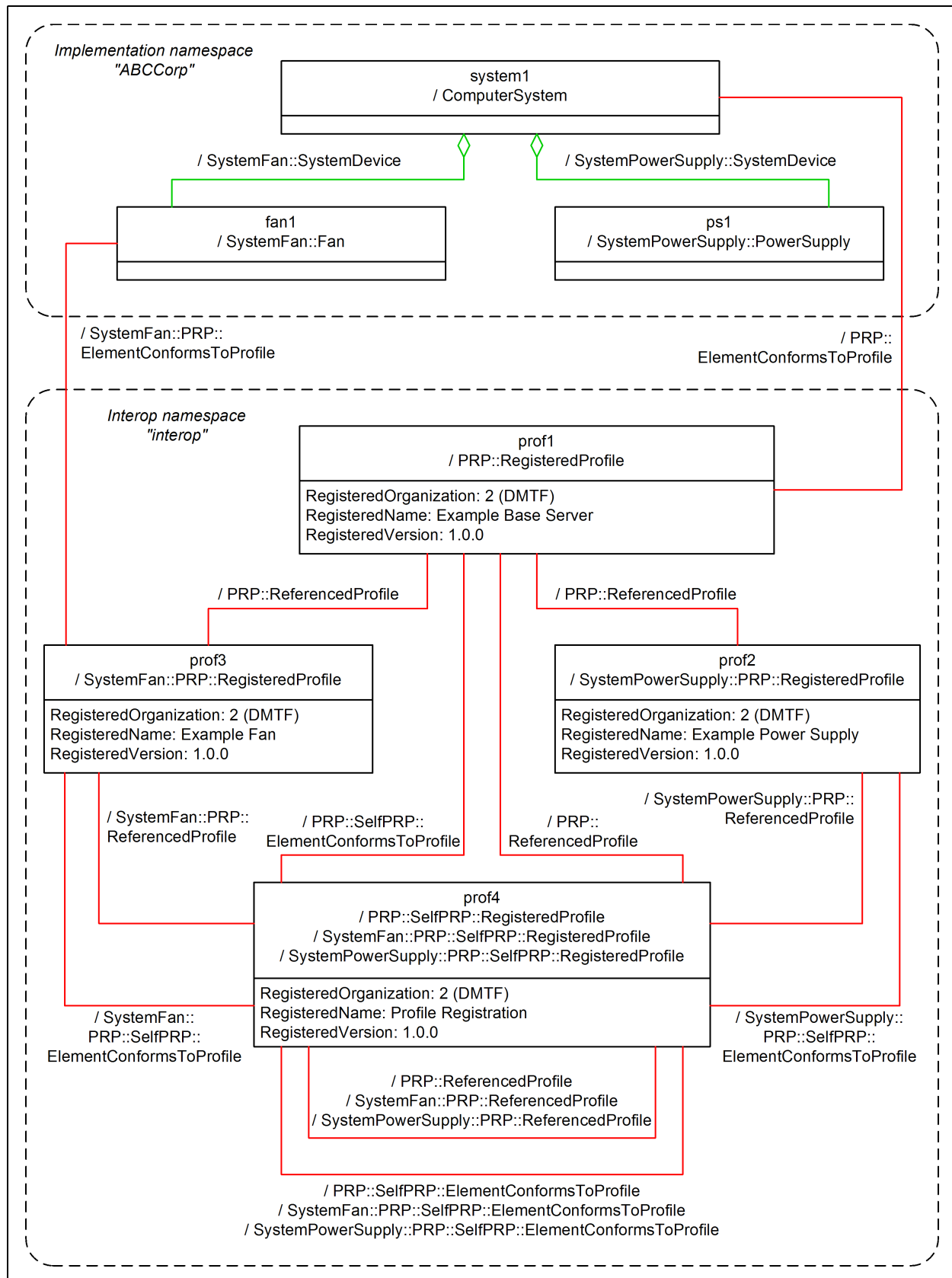


Figure 4 – Simple object diagram

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".

The Interop namespace contains four instances of `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).

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

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

Because some of the `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 `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 subclause "Cross-namespace associations" for more information about cross-namespace associations.

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

8.2 Use case: RetrieveProfileInformationForComputerSystem

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

This use case has the following preconditions:

- The instance path of a `ComputerSystem` instance (in the implementation namespace) is known.
- It is known that the Example Base Server profile is an autonomous profile and thus its implementation will support the central class methodology.

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

1. Invoke the `GetAssociatedInstancesWithPath()` for `ElementConformsToProfile` operation on that `ComputerSystem` instance. The resulting `RegisteredProfile` instances represent all profiles to which that `ComputerSystem` instance conforms.
2. Iterate through the retrieved `RegisteredProfile` instances and inspect their `RegisteredOrganization`, `RegisteredName` and `RegisteredVersion` property values, which identify the profiles to which the `ComputerSystem` instance conforms.

8.3 Use case: RetrieveProfileVersionForFan

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

This use case has the following preconditions:

- The instance path of a 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 sequence of steps:

1. Invoke the GetAssociatedInstancesWithPath operation on the given Fan instance, filtering on the ElementConformsToProfile association. This will retrieve all RegisteredProfile instances representing profiles to which that Fan instance conforms. In this scenario, only one 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 Fan instance conforms.

8.4 Use case: RetrieveProfileVersionForPowerSupply

This use case describes for the scenario defined in the SimpleStateDescription state description how a CIM client can retrieve the version of the Example Power Supply profile to which an instance of the PowerSupply adaptation conforms. In that scenario, the Example Power Supply profile (defining the PowerSupply adaptation) is a component profile and has been implemented using the scoping class methodology.

This use case has the following preconditions:

- The instance path of a 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 using the scoping class methodology.

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

1. Invoke the GetAssociatedInstancesWithPath operation on that PowerSupply instance, filtering on the SystemDevice association. This will retrieve the (one) ComputerSystem instance that is the scoping instance of the PowerSupply instance.
2. Invoke the GetAssociatedInstancesWithPath operation on that ComputerSystem instance, filtering on the ElementConformsToProfile association. This will retrieve all RegisteredProfile instances representing profiles to which that ComputerSystem instance conforms. In this scenario, only one instance representing the Example Base Server profile will be returned.
3. Invoke the GetAssociatedInstancesWithPath() for ReferencedProfile operation on the returned RegisteredProfile instance representing the Example Base Server profile. This will retrieve all 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 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 PowerSupply instance conforms.

8.5 Use case: AlgorithmForRetrievingProfileInformation

This use case describes for the general case 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.

This use case has the following preconditions:

- The instance path of a central instance of a given profile is known.
- 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.

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

1. Invoke the `GetAssociatedInstancesWithPath()` for `ElementConformsToProfile` operation on the central instance.
2. If this operation returns one or more `RegisteredProfile` instances, the profile has been implemented using the central class methodology, and each (typically one) returned instance represents a profile the central instance advertises conformance to.
 Their `RegisteredOrganization`, `RegisteredName` and `RegisteredVersion` properties of the returned instances identify these profiles.
3. If this operation returns no `RegisteredProfile` instances, the profile has been implemented using the scoping class methodology; in that case, follow these steps:
 - Navigate from the central instance to its scoping instance by following the scoping path defined in the profile.
 - Invoke the `GetAssociatedInstancesWithPath()` for `ElementConformsToProfile` operation on that scoping instance. This returns the `RegisteredProfile` instances representing the profiles to which the scoping instance advertises conformance.
 - If this operation returns one or more `RegisteredProfile` instances, the profiles of the scoping instance have been implemented using the central class methodology, and each (typically one) returned instance represents a profiles to which the scoping instance advertises conformance.
 Go to step 4.
 - If this operation returns no `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. Once that is reached, each (typically one) returned instance represents a profile to which the scoping instance advertises conformance.
 Go to step 4.
4. At this point, we know one (or more) `RegisteredProfile` instances representing profiles to which the top-most scoping instances advertise conformance.
 Select the profile of those top-most profiles that directly or indirectly references the profile you are interested in.
5. Invoke the `GetAssociatedInstancesWithPath()` for `ReferencedProfile` operation on the `RegisteredProfile` instance representing the selected top-most profile, 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.

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

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

8.6 Use case: DetermineConformingInstances

The following figure is an object diagram for this use case and illustrates an implementation of 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 implementation are the two Fan instances, `fan1` and `fan2`.

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.

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 using the `GetReferencingInstancesWithPath()` operation.

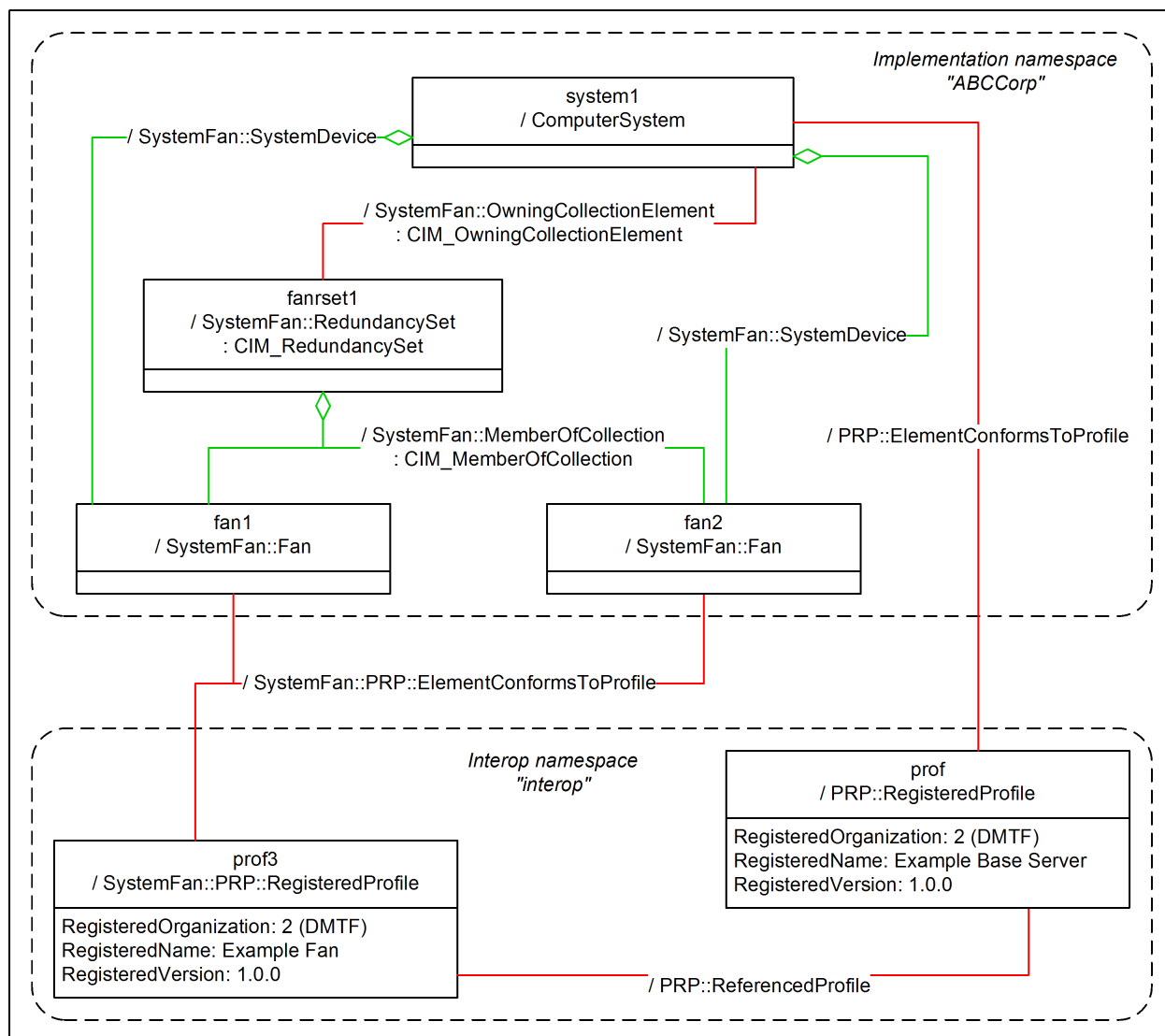


Figure 5 – Redundant fans object diagram

This use case has the following preconditions:

- The instance paths of all central instances of the Example Fan profile are known.
- The navigation graph between instances of all adaptations defined in the Example Fan profile is known.

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

1. For each central instance and for each association adaptation defined in the Example Fan profile that starts at the Fan adaptation, invoke the `GetAssociatedInstancesWithPath()` 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 RedundancySet instance `fanrset1` and the RegisteredProfile instance `profile2`.

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 ComputerSystem instance `system1`.

8.7 Use case: AlgorithmForDeterminingAdvertisedProfiles

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

This use case has the following preconditions:

- The namespace path of the Interop namespace of the WBEM server is known.

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

1. Invoke the `GetClassInstancesWithPath()` operation on the class of the `RegisteredProfile` adaptation in the Interop namespace.
This will retrieve the `RegisteredProfile` instances representing all profiles to which the WBEM server advertises conformance.
2. Iterate through these retrieved instances and inspect the values of their `RegisteredOrganization`, `RegisteredName` and `RegisteredVersion` properties, which identify these profiles.

8.8 Use case: AlgorithmForDeterminingTopLevelProfiles

This use case describes for the general case 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 that are implemented. This is accomplished by determining which instances of `RegisteredProfile` are not antecedents for any `ReferencedProfile` associations.

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 it is possible that autonomous profiles are referenced by other profiles.

This use case has the following preconditions:

- The namespace path of the Interop namespace of the WBEM server is known.

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

1. Invoke the `GetClassInstancesWithPath()` operation on the class of the `RegisteredProfile` adaptation in the Interop namespace.
This will retrieve the `RegisteredProfile` instances representing all profiles to which the WBEM server advertises conformance.
2. Invoke the `GetAssociatedInstancePaths()` operation on the class of the `RegisteredProfile` adaptation in the Interop namespace, filtering on the class of the `ReferencedProfile` association adaptation and on source role `Antecedent`.
This will retrieve the instance paths of the `RegisteredProfile` instances representing all profiles to which the WBEM server advertises conformance and that are referenced by other such profiles.
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.

8.9 Use case: AlgorithmForDeterminingCentralInstancesOfProfile

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.

This use case has the following preconditions:

- The namespace path of the Interop namespace of the WBEM server is known.
- The given profile is known by its registered name, organization and version.
- 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.

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

1. Invoke the `GetClassInstancesWithPath()` operation on the class of the `RegisteredProfile` adaptation in the Interop namespace.

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

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

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 `RegisteredProfile` instance for advertising conformance.

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

- Invoke the `GetAssociatedInstancesWithPath()` for `ElementConformsToProfile` operation on the `RegisteredProfile` instance.
- If one or more instances are returned, the profile has implemented the central class methodology.

Return from this recursive invocation of step 3.

- If no instances are returned, the profile has implemented the scoping class methodology.

Follow these steps:

- Invoke the `GetAssociatedInstancesWithPath()` for `ReferencedProfile` operation on the `RegisteredProfile` instance, filtering on the target role `Dependent`.

This will return the `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 `RegisteredProfile` instance representing the scoping profile of the profile.
- Navigate from the `RegisteredProfile` instance to its scoping instance by following the scoping path defined in the profile.

- Invoke the `GetAssociatedInstancesWithPath()` for `ElementConformsToProfile` operation on that scoping instance. This returns the `RegisteredProfile` instances representing the profiles to which the scoping instance advertises conformance.
- If this operation returns one or more `RegisteredProfile` instances, the profiles of the scoping instance have been implemented using the central class methodology, and each (typically one) returned instance represents a profiles to which the scoping instance advertises conformance.

Go to step 4.

- If this operation returns no `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. Once that is reached, each (typically one) returned instance represents a profile to which the scoping instance advertises conformance.

Go to step 4.

- At this point, we know one (or more) `RegisteredProfile` instances representing profiles to which the top-most scoping instances advertise conformance.

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

8.10 Use case: AlgorithmForDeterminingCentralOrScoping

This use case describes for the general case the algorithm for a CIM client to determine whether a profile represented by a given `RegisteredProfile` instance has been implemented using the central class methodology or the scoping class methodology.

This algorithm is based on whether `ElementConformsToProfile` associations are directly linked to the given instance of `RegisteredProfile`.

This use case has the following preconditions:

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

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

1. Invoke the `GetAssociatedInstancesWithPath()` for `ElementConformsToProfile` operation on the given `RegisteredProfile` instance.

If one or more instances are returned, the central class methodology has been implemented.

If no instances are returned, the scoping class methodology has been implemented.

If the profile represented by the given `RegisteredProfile` instance is an autonomous profile, the scoping class methodology also has been implemented at the same time, because for autonomous profiles, both advertisement methodologies fall together and result in the same implementation.

8.11 State description: PeerComponentProfileStateDescription

TBD: It is not clear what this scenario intends to say. Simple "peer component" profiles are shown already in the `SimpleStateDescription` scenario.

TBD: The description argues that this scenario supports the scoping class methodology, but both component profile implementations have used the central class methodology, so it is not clear why this argument is made.

The following figure illustrates the relationship between CIM_RegisteredProfile instances for the peer component profiles Example Fan and Example Sensor. The implementation and Interop Namespaces are depicted for illustrative purposes showing a typical implementation. See 7.1 for more information about namespaces.

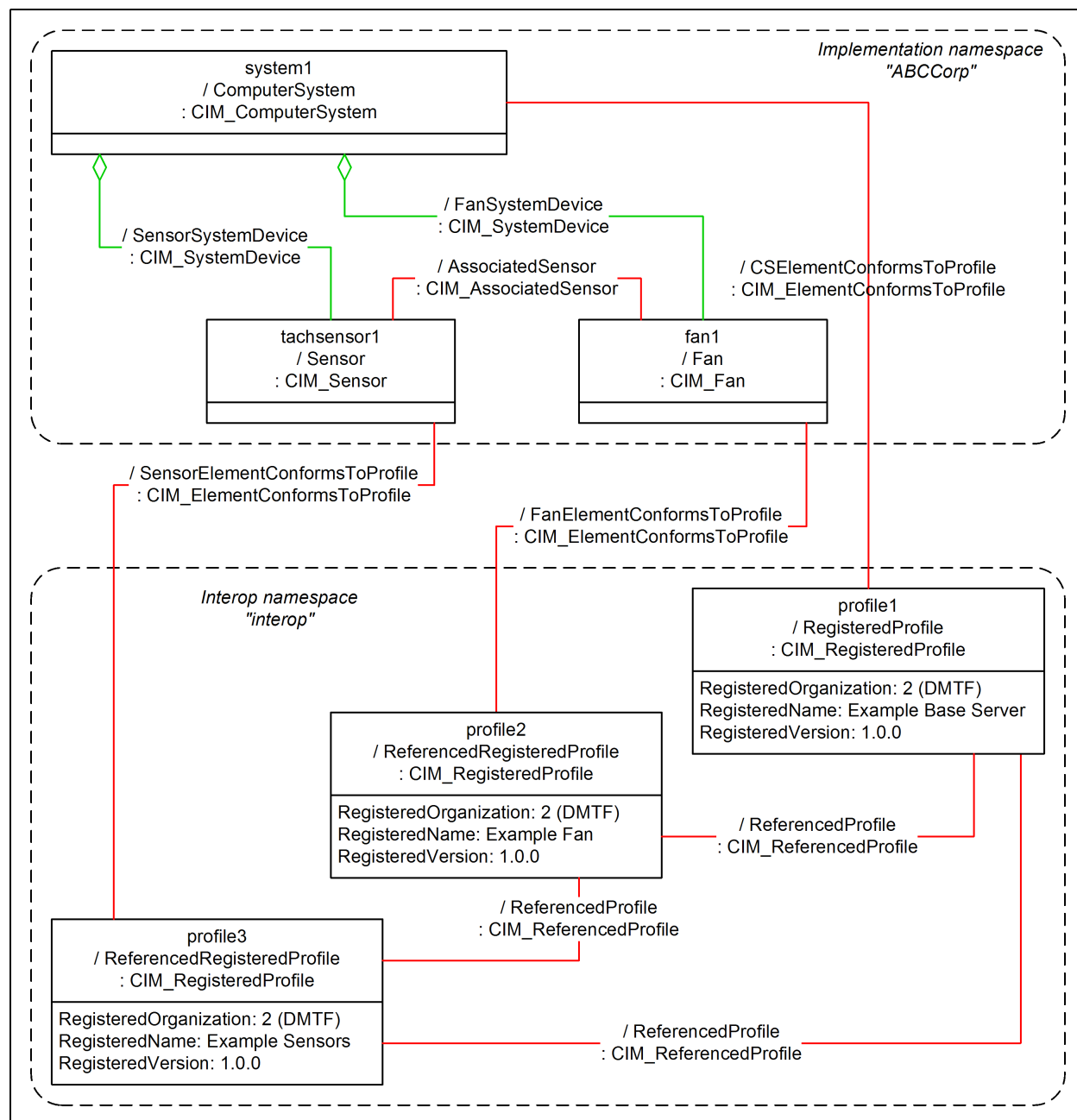


Figure 6 – Peer component profiles object diagram

In this figure, the Central Instances of three profiles are shown associated through CIM_ElementConformsToProfile to the instances of CIM_RegisteredProfile that represent the profiles with which they are compliant. Also represented is the CIM_RegisteredProfile hierarchy through the CIM_ReferencedProfile associations in the Interop Namespace. In this situation, the Base Server Profile (DSP1004) is the autonomous profile that references the component profiles, the Sensor Profile

(DSP1009) and the Fan Profile (DSP1013). This hierarchy would support the Scoping Class methodology for profile compliance advertisement. The relationship between peer component profiles, Fan and Sensor (that is, the Fan Profile includes the Sensor Profile and defines a tachometer sensor), is represented by an instance of the CIM_ReferencedProfile association.

8.12 State description: ProfileComplianceHierarchyStateDescription

The following figure depicts the hierarchy of RegisteredProfile instances associated through 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.

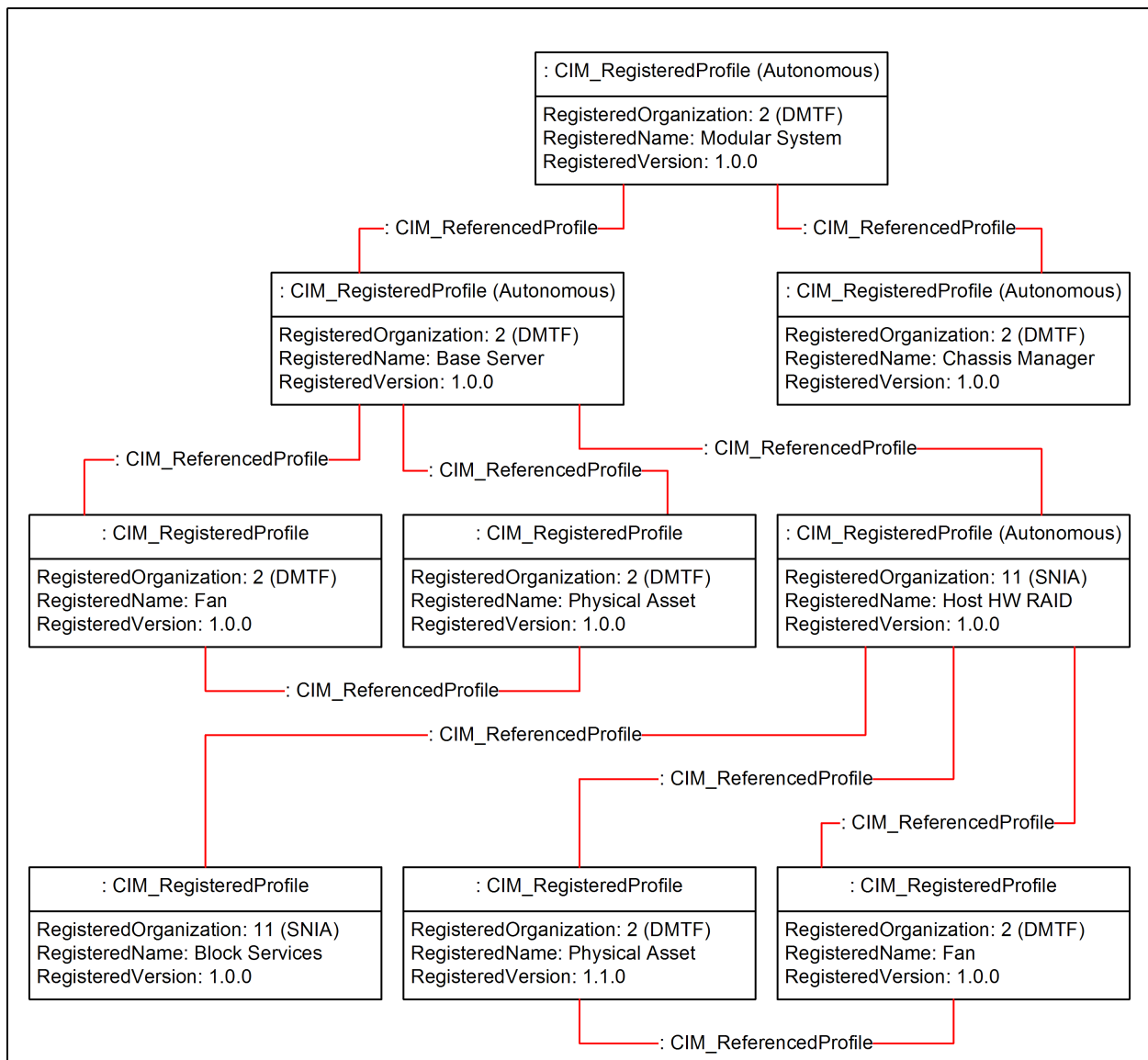


Figure 7 – Profile compliance hierarchy object diagram

ANNEX A

(informative)

Change log

Table 14 – Change log

Version	Date	Description
1.0.0 (prelim)	2006-12-06	Released as a Preliminary Standard
1.0.0	2007-06-25	Released as a Final Standard
1.1.0a	2011-12-19	<p>Released as Work in Progress, with the following changes:</p> <ul style="list-style-type: none"> • Converted to DMTF machine readable format. This included using new concepts from DSP1001 v1.1, such as class adaptations, features, constraints, generic operations and DMTF collaboration structure 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. • 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: <ul style="list-style-type: none"> • "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.

Bibliography

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

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