Document number: DSP1001

Date: 2009-11-04

Version: 1.1.0k

Management Profile Specification Usage Guide

Information for work in progress version:

This document is subject to change at any time without further notice.

It expires on: **2010-04-30**

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: E

16

17 Copyright Notice

18 Copyright © 2009 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

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

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33 withdrawn or modified after publication, and shall be indemnified and held harmless by any party imple-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

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

Contents

Figures

Tables

194

Foreword

The Management Profile Specification Usage Guide (DSP1001) was prepared by the DMTF Profile Infrastructure Working Group.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability.

Acknowledgements

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

- George Ericson, EMC
- Steve Hand, Symantec
- Jon Hass, Dell
- Michael Johanssen, IBM
- Andreas Maier, IBM
- Aaron Merkin, Dell
- Paul von Behren, Sun Microsystems

Document conventions

Any text in this document is in normal text font, with the following exceptions:

- References to clause names use normal text font; if they consist of more than one word, the clause name is quoted using double quotes, such as in "CIM elements".
- Important terms that are used for the first time are marked in *italics*.
- The usage of terms link to the term definition defined in the "Terms and definitions" clause, enabling easy navigation to the term definition.
- ABNF rules are in `monospaced font`.

Format definitions in this document are specified using ABNF (see RFC5234), with the following deviations:

- Literal strings are to be interpreted as case-sensitive Unicode characters, as opposed to the definition in RFC5234 that interprets literal strings as case-insensitve US-ASCII characters.

Management Profile Specification Usage Guide

1 Scope

This guide defines the usage of and requirements for management profiles and management profile specification documents.

A *management profile* (short: *profile*) defines a management interface between implementations of a WBEM service and a WBEM client. In addition, a profile may define a management interface between a WBEM service and a WBEM listener for the delivery of indications. The management interfaces establish a contract between the involved WBEM components, but are not an API because they do not define a programming interface. A profile defines a model and its behavior in the context of a management domain. Model and behavior are defined by selecting, specializing and sometimes constraining elements from a schema and the set of operations and indications[1] for a particular purpose. A profile establishes a relationship between the model and the management domain. A profile defines use-cases on the model that illustrate client visible behavior.

A *management profile specification* document (short: *profile specification*) contains the textual specification of one or more management profiles and may also contain content that does not specify a profile.

Profiles and profile specifications may be owned by DMTF or by other organizations.

The audience for this guide is anyone creating profiles or profile specifications (regardless of whether these are published by DMTF or published by other organizations), and implementers of profiles.

NOTE This guide is not a template for a profile specification. To create a profile specification, start with the publishing organization's template and add clauses as described in this guide. For profiles published by DMTF, use DSP1000.

NOTE This guide is not a profile specification; it defines the requirements for creating profiles or profile specifications.

2 Normative references

The following referenced documents are indispensable for the application of this guide. For dated or versioned references, only the edition cited (including any corrigenda or DMTF update versions) applies. For undated and unversioned references, 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 DSP1033, *Profile Registration Profile 1.0*
http://www.dmtf.org/standards/published_documents/DSP1033_1.0.pdf

DMTF DSP1054, *Indications Profile 1.0*
http://www.dmtf.org/standards/published_documents/DSP1054_1.0.pdf

DMTF DSP0200, *CIM Operations over HTTP 1.3*
http://www.dmtf.org/standards/published_documents/DSP0200_1.3.pdf

DMTF DSP0215, *Server Management Managed Element Addressing Specification 1.0*
http://www.dmtf.org/standards/published_documents/DSP0215_1.0.pdf

[1] In this case the term indication is used as an interaction between a WBEM service and a WBEM listener.

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

DMTF DSP0228, *Message Registry XML Schema 1.0*
http://www.dmtf.org/standards/published_documents/DSP0228_1.0.xsd

DMTF DSP4004, *DMTF Release Process 2.0*
http://www.dmtf.org/standards/published_documents/DSP4004_2.0.pdf

DMTF DSP8016, *WBEM Operations Message Registry 1.0*
http://schemas.dmtf.org/wbem/messageregistry/1/dsp8016_1.0.xml

DMTF DSP8020, *Message Registry XML Schema Specifiation 1.0*
http://www.dmtf.org/standards/published_documents/DSP8020_1.0.xsd

IETF RFC3629, *UTF-8, a transformation format of ISO 10646*, November 2003
http://tools.ietf.org/html/rfc3629

IETF RFC5234, *ABNF: Augmented BNF for Syntax Specifications*, January 2008
http://tools.ietf.org/html/rfc5234

ISO/IEC Directives, Part2:2004, *Rules for the structure and drafting of International Standards*
http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype

OMG UML Superstructure, *OMG Unified Modeling Language (OMG UML) Superstructure 2.1.2*

Regular Expressions, in The Open Group, *The Single UNIX ® Specification, Version 2*
http://www.opengroup.org/onlinepubs/7908799/xbd/re.html

3 Terms and definitions

In this guide, 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.

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 guide, clauses, subclauses or annexes indicated with "(informative)" as well as notes and examples do not contain normative content.

The terms defined in DSP0004 and DSP0223 apply to this guide.

3.2

abstract profile

a special kind of profile specifying common elements and behavior as a base for derived profiles. For a complete definition, see 7.6.3.11 .

3.3

adaptation

short form for class adaptation.

3.4
adaptation instance
an instance of an adapted class that complies with all requirements of the class adaptation. For details see 5.3 .

3.5
adapted class
a class that is the subject of a class adaptation. For details, see 7.10 .

3.6
autonomous profile
a profile that addresses an autonomous and self-contained management domain. For details, see 7.6.5.2 .

3.7
backward compatibility
a characteristic of profiles enabling clients written against prior minor versions of a profile to use the functionality specified by that version in context of an implementation of a later minor version, without requiring modifications of the client; for a complete definition, see 7.14 .

3.8
base adaptation
a class adaptation that is used as the base for another class adaptation; for details, see 7.10.2.2 .

3.9
base profile
a profile that is derived by one or more other profiles. For a details, see 7.6.2 and 7.6.3 .

3.10
central class adaptation
a specifically designated class adaptation in a subject profile. The central class adaptation is the focal point of the subject profile. For a complete definition, see 7.6.5.4 .

3.11
schema element
generally, refers to schema elements as defined in DSP0004. In this guide, the term is used for the subset of schema elements that may be constrained by profiles: classes (including associations and indications), properties (including references), methods, and parameters.

3.12
class
if used without qualification this term refers to a CIM class that may also be an association or an indication. To refer to a CIM class that is not an association or an indication, use the term "ordinary class". For a complete definition, see DSP0004.

3.13
class adaptation
a named profile element that defines requirements and constraints on a class. A class adaptation adapts a class definition from a schema for a particular purpose and may be based on other class adaptations. For a complete definition, see 7.10 .

3.14
client
in this guide, a WBEM client or a WBEM listener that acts as a consumer of an implementation. See also the term "implementation".

3.15
component profile
a profile that addresses a subset of a management domain. For details, see 7.6.5.3 .

concrete profile
any profile that is not an abstract profile. For a complete definition, see 7.6.4.2 .

3.16
conditional
a requirement level designating profile elements or referenced profiles if their implementation is required under specified conditions, with no deviation permitted.

3.17
conditional profile
a referenced profile that is referenced with the conditional requirement level.

3.18
deprecated
keyword indicating that a profile element or profile defined behavior is outdated and has been replaced by newer constructs. For details see 7.14 .

3.19
derived profile
a profile that is based on one or more other profiles. For a complete definition, see 7.6.3 .

3.20
effective class adaptation
a class adaptation constructed at implementation time that in context of an implementation profile set merges all implementation requirements from all base adaptations. For details, see 8.1.2 .

3.21
event
the occurrence of a phenomenon of interest to a client. For details, see 6.8 .

3.22
existence of instances
an adaptation instance exists in a namespace if it is observable by clients. Existance of an instance does not imply a physical representation of the instance such as for example a record in a database.

3.23
exposed property or method
a property or method that is available to clients using an adaptation. The set of properties or methods exposed by an adaptation is the union of all properties or methods defined in the adapted class and its superclasses. In the case where a property or method overrides a property or method defined in a superclass, the combined effects are exposed as a single property or method.

3.24
feature
a profile element that groups the decisions for the implementation of one or more profile elements into a single decision. This grouping is established by defining the implementation of other profile elements dependent on the implementation of the feature. For a complete definition, see 7.12 .

3.25
implementation
synonym for profile implementation

3.26
implementation adaptation set
the set of effective class adaptations to be implemented in context of an implementation profile set. For details, see 8.1.2 .

3.27
implementation-required
a phrase indicating that the implementation of a profile element is required within an implementation of one or more profiles, including the case where an optional profile element was selected to be implemented. For details, see 8.1.2 .

3.28
implementation profile set
the set of profiles that is implemented. For details, see 8.1.2 .

3.29
incompatibility
a change that breaks backward compatibility.

3.30
management domain
area of work or field of activity with common management requirements, common terminology, and related management functionality. For details, see 6.2 .

3.31
managed environment
a concrete occurrence of the management domain. A managed environment is composed of managed objects. For details, see 6.4 .

3.32
managed object
a physical entity, a service, or other kind of resource that exists independently of its use in management. Managed objects exist in managed environments. For details, see 6.4 .

3.33
managed object type
a conceptual generalization or type of manageable things. For details, see 6.3 .

3.34
management profile
definition of a management interface between a WBEM service and a WBEM client or a WBEM listener. For a complete definition, see clause 1 .

3.35
management profile specification
a specification document that contains the textual specification of one or more management profiles and optionally content that does not represent a management profile. For a complete definition, see clause 1 .

3.36
mandatory
a requirement level designating profile elements if their implementation is strictly required and no deviation is permitted.

3.37
mandatory profile
a referenced profile that is referenced with the mandatory requirement level.

3.38
match
keyword indicating that a property or parameter value is within the values specified by a pattern. For details see 9.2.5 .

3.39
method requirement
a requirement stated as part of a class adaptation that defines requirements and constraints on a method exposed by the adapted class. For details, see 7.10.3.1 .

3.40
message registry
a published repository of messages formatted as defined in DSP0228.

3.41
metrics registry
a published repository of metric definitions, and optionally statistics definitions, formatted as defined in DSP8020.

3.42
named profile element
a profile element that is assigned a name with profile name scope. For details, see 7.3 .

3.43
operation requirement
a requirement stated as part of a class adaptation that defines requirements and constraints on an operation defined in an operation specification. For details, see 7.10.3.2 .

3.44
optional
a requirement level designating profile elements or referenced profiles if their implementation is at the choice of the implementation, with no implied preference.

3.45
optional profile
a referenced profile that is referenced with the optional requirement level.

3.46
ordinary class
a class that is not an association or an indication. For a complete definition, see DSP0004.

3.47
organization
in this guide, refers to a consortium, standards group, or company creating a management profile.

3.48
pattern
specification of the permissible values for a property or parameter. See also the term "match". For details see 9.2.5 .

3.49
profile
synonym for management profile, see 3.34 . For a complete definition, see clause 1 .

3.50
profile defined model
a model of a management domain (or a subset of a management domain) defined by a profile that is composed of class adaptations. For details, see 6.1 .

3.51
profile document
synonym for management profile specification. It is preferred to use the term profile specification instead.

3.52
profile element
an event, a feature, a profile reference, a registry reference, an adaptation, a method requirement, an operation requirement or a property requirement.

3.53
profile implementation
the WBEM service side realization of a profile. For example, in server side infrastructures using CIM providers, profile implementation refers to the CIMOM and the set of providers implementing the profile.

3.54
profile specification
synonym for management profile specification, see 3.35 . For a complete definition see clause 1 .

3.55
profile reference
a profile element that references another profile. For details, see 7.6.2 .

3.56
property requirement
a requirement stated as part of a class adaptation that defines requirements and constraints on a property exposed by the adapted class. For details, see 7.10.2.5 .

3.57
referenced profile
a profile that is referenced by a profile with a requirement level. For a complete definition, see 7.6 .

3.58
referencing profile
a profile that references another profile. For a complete definition, see 7.6 .

3.59
registry reference
a profile element referencing message registry or a metrics registry. For details, see 7.8 .

3.60
related profile
a referenced profile or a base profile.

3.61
requirement level
designator that indicates the requirement for implementing profile elements or referenced profiles.

3.62
schema
a named set of classes with a single defining authority or owning organization. The classes in a schema have the same schema prefix in their class name. For a complete definition, see DSP0004.

NOTE DMTF defines two schemas: The Common Information Model (schema prefix CIM) and the Problem Resolution Schema (schema prefix PRS)

3.63
scoping class adaptation
a specifically designated class adaptation in a subject profile that is the algorithmic focal point for identifying profile conformance when using the scoping class methodology. For a complete definition, see 7.6.5.5 .

3.64
scoped profile
a profile that receives a scope provided by a scoping profile. Synonymous to component profile. For details, see 7.6.5 .

3.65
scoping path
an association traversal path between the central class adaptation and the scoping class adaptation. For details, see 7.6.5.6 .

3.66
scoping profile
a profile that provides a scope to a scoped profile by defining a class adaptation that is compatible with the scoping class adaptation defined by a scoped profile. For details, see 7.6.5 .

3.67
span of a class adaptation
the directed acyclic graph that contains the class adaptation, all (direct or indirect) base adaptations of the class adaptation, the adapted class and all its superclasses. For a complete definition, see 7.10.2.2 .

3.68
subject profile
a profile created or verified in conformance to this guide.

3.69
trivial class adaptation
a class adaptation that does not add requirements beyond those defined by the adapted class and, if defined, by its base adaptations. For details, see 9.4.7.4 .

3.70
WBEM client
a CIM client (see DSP0004) that supports a WBEM protocol. For details, see DSP0223..

3.71
WBEM listener
a CIM listener (see DSP0004) that supports a WBEM protocol.

3.72
WBEM protocol
a communications protocol between WBEM client, WBEM service and WBEM listener. A WBEM protocol defines how the WBEM operations and WBEM indications work, on top of an underlying protocol layer (for example, HTTP, SOAP, or TCP). For details, see DSP0223.

3.73
WBEM service
a CIM service (see DSP0004) that supports a WBEM protocol. For details, see DSP0223.

4 Symbols and abbreviated terms

Most of these symbols and abbreviated terms are applicable also to profile specifications.

NOTE A list of symbols and abbreviated terms to be included in profile specifications is provided in DSP1000.

For the purposes of this guide, the following symbols and abbreviated terms apply, in addition to those defined in DSP0004 and DSP0223:

4.1
UFcT
User Friendly class Tag, as defined in DSP0215.

4.2
UFiT
User Friendly instance Tag, as defined in DSP0215.

4.3
CSD
DMTF collaboration structure diagram. For details, see 9.2.2.3 .

5 Conformance

This clause defines conformance requirements for profiles, profile specifications, implementations and instances.

5.1 Profile and profile specification conformance

A profile is conformant to this guide if it satisfies all normative requirements defined in this guide for profiles. The normative requirements for profiles are detailed in clause 7 .

A profile specification is conformant to this guide if it satisfies all normative requirements defined in this guide for profile specifications. The normative requirements for profile specifications are detailed in clause 9 .

5.2 Implementation conformance

5.2.1 Interface implementation conformance

An implementation of a profile is interface conformant to the profile if it conforms to all profile requirements that are defined only in terms of the profile defined model. Interface implementation conformance does not cover the relationship of instances and managed objects.

Interface conformance can be validated exclusively by the use of the profile defined interface; this validation approach is also refered to as black box testing.

Requirements only defined in terms of the model are for example:

- Value constraints that restrict a property value to a set of possible values, such as for example restricting the value of an EnabledState property to the values 2 (Enabled) or 3 (Disabled).
- Requirements for the existence of instances as a result of the successful execution of an operation or method

Requirements that are not only defined in terms of the model are for example:

- The requirement that specific managed objects are to be represented by instances
- The requirement that a property value shall reflect a part of the state of a managed object, such as for example stating that the value 2 (Enabled) of an EnabledState property corresponds to the On state of the managed object.

- The requirement that the execution of an operation or method causes a specified change in the managed environment, such as for example the activation of a managed object in the case where a change of the EnabledState property to 2 (Enabled) is requested.

5.2.2 Full implementation conformance

Full implementation comformance extends interface implementation conformance by also taking into account profile defined requirements that also take into consideration the managed environment. For example, such requirements establish the relationship of the profile defined model and the managed environment.

Functional conformance can only be validated by crosschecking the situation in the managed environment with the situation as viewed through the profile defined interface. Functional conformance validation requires direct access to the managed environment such that the situation inspected through that direct access can be cross checked against the situation presented by an implementation through the profile defined model; this validation approach is also referred to as white box testing.

5.2.3 Implementation conformance of multiple profiles

An implementation that implements multiple profiles is conformant to that set of profiles, if it is conformant to each profile.

NOTE Profiles may have dependencies such as for example class adaptations in one profile being based on class adaptations in other profiles.

5.2.4 Implementation conformance of profile versions

Profile versions are identified with the complete set of version numbers as defined in DSP4004: Major, minor and update version number. However, as defined in 7.6.2, a subject profile refers to related profiles by specifying only the major and minor version number, implying the latest published update versions of the related profiles. Consequently it is possible that various implementations of a comprehensive set of profiles (such an identified version of a particular subject profile and all its related profiles) that are created at different points in time use different update versions of the related profiles.

As a consequence, conformance of a profile implementation to a profile is only defined with regard to a specific update version of that profile.

For example, if a particular profile P1 was written when version 1.0.1 of a referenced profile P2 was published, P2 is referenced as version 1.0 in the related profiles table of P1. At this point in time, an implementation of P1 and P2 would implement version 1.0.1 of P2. Once version 1.0.2 of P2 is published, that version should now be implemented by any new implementation of P1 and P2. In this case the first implementation conforms to version 1.0.1 of P2, and the second implementation conforms to version 1.0.2 of P2. In addition, the backward compatibility rules defined in 7.14 strive for only permitting changes that do not invalidate of the second implementation to version 1.0.1 of P2; however – as outlined before - it cannot be completely ruled out that version 1.0.2 introduces incompatible changes; for details, see 7.14 .

5.2.5 Client implementation conformance

There is no explicit concept of client conformance. However, a client intending to successfully interoperate with an implementation needs to adhere to the preconditions defined by the profile and by other specifications referenced by the profile that the implementation conforms to.

5.3 Instance conformance

An instance of a CIM class is conformant to a class adaptation if it satisfies all normative requirements of the class adaptation, including those originating from base adaptations and from the schema.

NOTE The collection of normative requirements of a particular class adaptation in context of the implementation of a set of profiles is a complex process considering all involved sources of requirements such as base

650 adaptations defined in referenced profiles, the schema and operation specifications; see 8.1.2 for a de-
651 tailed description of that process.

652 5.4 DMTF conformance requirements

653 The following rules apply to management profiles and management profile specifications owned by
654 DMTF:

655 • Management profiles owned by DMTF shall conform to this guide. The normative requirements
656 for profiles are detailed in clause 7 .

657 • Management profile specifications owned by DMTF shall conform to this guide. The normative
658 requirements for profile specifications are detailed in clause 9 . In addition, the standard DMTF
659 specification format (see DSP1000) applies to DMTF owned management profile specifications.

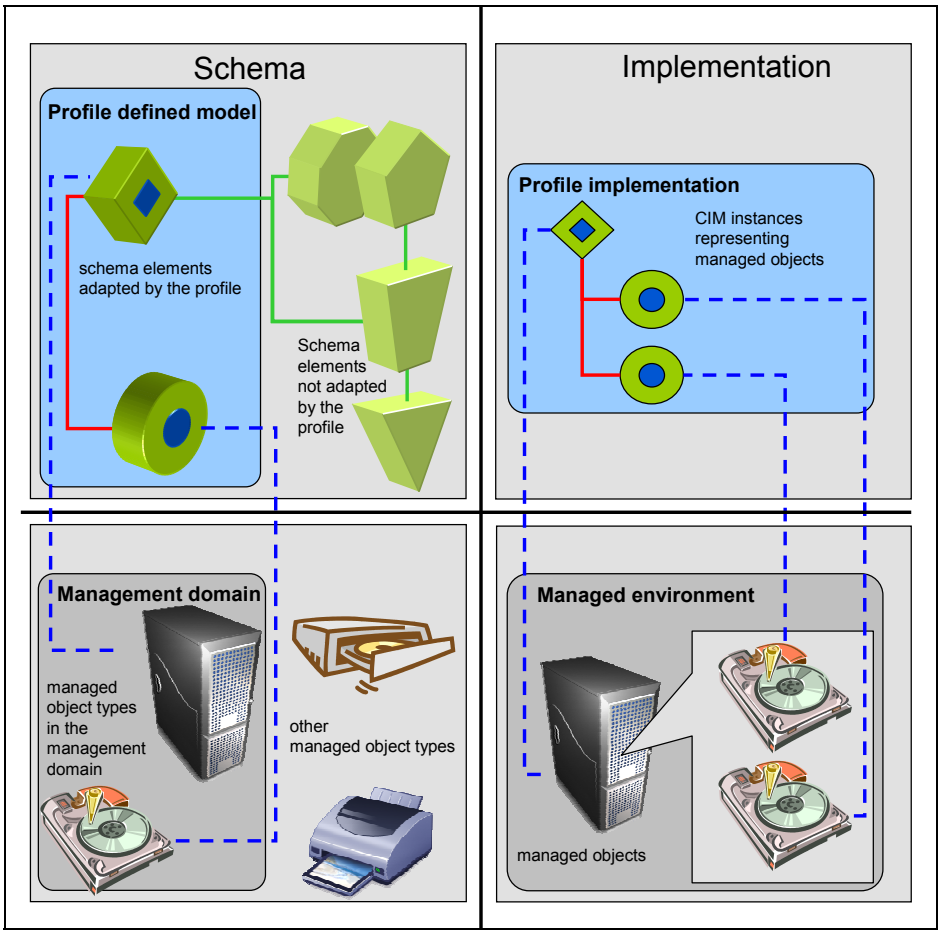
660 NOTE Other organizations may create their own guidelines for management profile specifications they publish. If
661 such profile specifications are to be conformant to this guide, these guidelines would have to incorporate
662 or reference, and optionally extend the requirements defined in this guide.

663 6 Concepts

664 This clause presents an informative introduction to general profile concepts established by this guide.

665 6.1 Overview

666 Figure 1 illustrates the profile defined model and its relationship to the management domain, as well as a
667 related profile implementation and its relationship to a managed environment.



668

669 **Figure 1 – Profile and management domain**

670 The left side of Figure 1 shows the profile defined model and its related management domain. Model and
671 behavior are defined by selecting, specializing and sometimes constraining elements from a schema and
672 the set of operations for a particular purpose; in other words, the profile adapts elements from a schema
673 for a particular purpose. The management domain is composed of managed object types. The classes
674 adapted by a profile model aspects of these object types. A profile establishes a relationship between the
675 model and the management domain. In addition, a profile defines use-cases on the model that illustrate
676 client visible behavior.

677 The right side of Figure 1 shows a profile implementation and a related managed environment. Each
678 profile implementation provides access to a set of related CIM instances to a CIM client. These CIM
679 instances represent corresponding managed objects in the managed environment and conform to the
680 client visible management interfaces and behaviors defined in the profile. Note that the right side of
681 Figure 1 shows only one profile implementation and only one related managed environment; however, in
682 reality potentially multiple implementations of a profile coexist, and each profile implementation typically
683 provides management capabilities for multiple related managed environments.

684 6.2 Management domain

685 A profile describes a *management domain* by defining the set of *managed object types* that compose the
686 management domain. In addition, the profile may define requirements and constraints on the components
687 of the management domain.

688 A management domain is area of work or field of activity. Commonalities in a management domain are a
689 set of common management requirements, a common terminology, and related functionality. Examples
690 for management domains are computer system, system virtualization or file system.

691 Complex management domains may be subdivided into smaller management domains where each
692 subdomain narrows down the area of work or field of activity. For example a subdivision of the file system
693 management domain might contain management subdomains such as file access, file locking or file
694 representation.

695 If a management domain is subdivided into a set of subdomains, these may be likewise covered by
696 separate profiles. This guide defines several types of profile relationships enabling this decomposition.

697 6.3 Managed object type

698 An *managed object type* is a conceptual generalization or type of manageable things in a management
699 domain. Examples for managed object types composing the computer system management domain are
700 system, device or service. Examples for managed object types composing the file system management
701 domain are file, directory, access list or lock.

702 Relationships may exist between managed object types. For example, in the file system management
703 domain directories are composed out of files, and files may be linked to each other.

704 6.4 Managed environment and managed objects

705 A *managed environment* is a concrete occurrence of a management domain, and is composed of
706 *managed objects*. For example, a managed environment within the file system management domain is a
707 concrete Linux ext3 file system that resides on some storage media, and is composed of objects such as
708 the file system itself, its files, directories, links, access lists or quotas. For a particular type of managed
709 environment (such as for example Linux ext3 file systems) specific management instrumentation (such as
710 a set of commands, or an API) may exist that allow the inspection and manipulation of managed objects
711 in respective managed environments. For example, instances of the Linux ext3 filesystem in a desktop
712 installation may be inspected and manipulated through means of the Linux ext3 file system device
713 drivers.

Profiles are implemented for one or more types of managed environments. For example, for a profile addressing the file system management domain one implementation might cover the Linux ext3 file system and another separate implementation might cover the FAT file system and the Microsoft NTFS file system.

6.5 Profile definition

A profile defines a management interface for a management domain. The semantics of that management interface as well as the behavior of the managed objects in their managed environment are defined by a model that is composed of a set of class adaptations. Each class adaptation defines a set of requirements and constraints on the use of a class for a particular purpose. Class adaptations are defined in 7.10 .

6.6 Relationships between profile definition and management domain

A profile defines the following mappings:

- the mapping of managed object types composing a management domain to CIM classes
- the mapping of managed objects composing a managed environment to CIM instances

These mappings have a substantial impact on the applicability of the profile and should be stated with great care, particularly when specifying the exact set or subset of managed objects that are to be represented by CIM instances.

In a managed environment the managed objects or relationships between them potentially appear, disappear or change at any time.

For example in a file system files are frequently created, deleted or modified. Such changes may be effected by means of the management interface defined by the profile as described in 6.7, but in general the cause for such changes is outside of the scope of the implementation of a profile.

Profiles define the lifecycle of CIM instances and how each CIM instance represents an aspect of a managed object in a managed environment. Profiles do not define a data representation for CIM instances, or a physical representation of the managed objects that they represent. The lifecycle of a CIM instance defines the conditions under which each CIM instance exists. Existence of a CIM instance means that the instance is visible to clients; existence does *not* imply or define a permanent physical representation of the CIM instance (like for example an entry in a management database). Instead, a CIM instance is considered a formalized view of an aspect of a managed object that is part of the managed environment.

Consequently, the only cause for a change in a CIM instance is a respective change in the managed object represented by the instance. It is emphasized that this is also the case if the change was caused by the execution of a method on a CIM instance that represents that managed object. In this case the execution of the method needs to effect the change on the managed object first, and only as a result of that change is the CIM instance changed.

CIM instances represent managed objects in the managed environment. This includes reflecting the state of the managed object after completing changes effected through the model, such as the invocation of methods (see 7.10.3.1) or operations (see 7.10.3.2). However, after, or coincident with such a change, other actions not effected through the model can also affect the state and are represented by the CIM instance. This situation drives the need for profiles to define the means that indicate completion for model effected changes.

CIM instances are inherently volatile. A profile intending to enable a client to continuously monitor the state of a managed object existing in a managed environment has two possibilities:

- require the client to continuously poll the information from the implementation. In this situation the client could repeatedly invoke the GetInstance( ) operation of the CIM instance representing the specific aspect being monitored. In a more comfortable case the profile could adapt a class providing a specific method designed to return information about any changes since the last poll.

- model indications as described in 6.8

NOTE If an implementation of a profile binds keys of CIM instances to attributes of managed objects that can be renamed or in cases where previously used values can be reused for new instances,, a polling client inspecting these instances may not be able to detect deletion or removal of the object represented by that instance and its subsequent replacement by another object represented by a CIM instance with an identical key.

A profile establishes the relationship between objects in the managed environment and their representation in the profile defined model through definitions such as:

- Definition of the mapping of the object in the managed environment to one or more CIM instances. This defines the set of CIM instances that represent aspects of the managed object. In many cases this is a one to one mapping, but complex managed objects often require more than one CIM instance for their representation, with each instance addressing an aspect of the object.
- Definition of the mapping of aspects of the state of the managed object to property values in the CIM instance.
- Definition of the mapping of relationships between managed objects in a managed environment and the representation of these relationships through instances of CIM associations.

6.7 Model effected control of managed objects in a managed environment

CIM based control of managed objects in a managed environment is requested through invocations of methods or operations. Methods might be defined on a CIM instance that represents an aspect of the managed object in the managed environment, or might be defined on CIM instances representing other entities such as for example services. The profile implementation issues requests to the managed object in the managed environment in order to perform the defined semantics of the method or operation. The mechanisms applied for this forwarding are implementation dependent. Depending on conditions that prevail in the managed environment the request may or may not succeed.

6.8 Events and indications

An event is an observable occurrence of a phenomenon in a managed environment. Events are not published in CIM directly; instead, notifications about an event are published by means of CIM indications. Profiles define events of interest to clients and the indications reporting the events. Conceptually, events occur in scope of the managed environment or in scope of CIM model; however note that any change in the CIM model is only the result of a preceding corresponding change in the managed environment. Notifications of events that map onto changes in a CIM instance are typically modeled as lifecycle indications. Other event notifications are typically modeled through process indications.

Lifecycle indications report the following events on CIM instances:

- Creation
- Deletion
- Read operation
- Modification
- Invocation of methods

Lifecycle indications enable a client to detect and / or continuously monitor changes in the managed environment through the observation of model changes. Recall that CIM instances represent aspects of managed objects in managed environments. Lifecycle indications do not necessarily report the creation or deletion of respective managed objects in the managed environment; instead the lifecycle of CIM instances is defined by profiles, and is ultimately controlled by the implementation. A profile may specify conditions under which a CIM instance is created. For example, a profile addressing the management domain of file management might specify that each file system is represented by an instance of the

CIM_FileSystem class. In this case, the lifecycle of the CIM instance is directly coupled to that of the managed object. However, note that for example in context of mounted filesystems there may be situations such as a loss of network connectivity, where an implementation is unable to obtain the complete state required to for the existence of the CIM instance in a namespace. In this case the implementation would have to indicate a communication error, leaving the client uncertain about the state of the filesystem.

On the other hand, if the profile had specified that only *mounted* file systems are represented by an instance of the CIM_FileSystem class, then the CIM instance would represent the filesystem only while it is mounted, and not reflect the lifecycle of the filesystem itself; that is, the CIM instance would be created whenever the filesystem is mounted and destroyed whenever the filesystem is unmounted, each time requiring respective lifecycle indications.

Process indications report events occurring either within the CIM model or within the managed environment. For example, a profile covering the file management domain might specify an alert indication that reports that a disk space threshold is exceeded.

7 Profile definitions

This clause defines the requirements for definitions in profiles. It focuses on the profile content, regardless of the format that is chosen to specify the profile. Clause 9 defines the requirements for profile specification documents, focusing on formal document aspects.

7.1 Usage of requirement levels

This subclause define the usage of requirement levels by profiles. Requirement levels designate the requirement for implementing profile elements.

The following requirement levels are defined:

- Mandatory, as defined in 3.36
- Optional, as defined in 3.44
- Conditional, as defined in 3.16

NOTE Requirement levels are formally defined only for the designation of profile elements including profile references. However, profiles may state other provisions such as for example instance requirements or indication generation requirements using normative language (e.g. "shall", "may", "should", etc.).

7.1.1 Usage of the "mandatory" requirement level

Profiles shall designate a profile element as mandatory if the functionality defined by the profile element is required for a functioning implementation of the profile.

7.1.2 Usage of the "optional" requirement level

Profiles shall designate a profile element as optional if the functionality defined by the profile element is not required, but is auxiliary or complementary for a functioning implementation of the profile.

Profiles should specify a mechanism that allows clients to detect whether or not an optional profile element is implemented.

A profile that intends to define multiple optional elements that are useful to clients only as a group should define an optional feature (see 7.12), and define the elements as conditional on the implementation of that optional feature.

7.1.3 Usage of the "conditional" requirement level

Profiles shall designate a profile element as conditional if the functionality defined by the profile element is required only under certain conditions, but is optional otherwise. For any definition designated as conditional, the condition shall be defined using one of the mechanisms defined in 7.2.

7.2 Definition of conditions

This subclause defines mechanisms for the definition of the condition that determines whether a conditional profile element must be implemented.

7.2.1 General

As defined in 7.1.3, profiles shall define a condition for any conditional elements. Profiles shall apply only the mechanisms defined in 7.2 defining such conditions. Subclauses 7.2.2 to 7.2.7 define basic types of conditions. Complex conditions may expressed as combinations of these basic condition types using the Boolean operators AND, OR, NOT, XOR, IMPLIES, ANY and ALL.

Some of these mechanisms are deprecated. New profiles and revisions of existing profiles should not use such deprecated mechanisms.

NOTE Conditions control conditional implementation requirements. Conditions are resolved at implementation time and are complied with by implementers as they implement conditional elements in the case where the condition is true. Conditions themselves are not generally directly observable by clients of profile implementations; however, the effect of implementing conditional elements is observable by clients.

NOTE Conditions are not to be confused with implementation decisions made by profile implementers. A condition does not need to be based on such decisions. For example, a condition may be tied to circumstances in the type of management environment addressed by an implementation, not leaving any room for a decision to be made.

7.2.2 Profile implementation condition

A profile may specify a condition based on the implementation of a referenced profile. This kind of condition is called a *profile implementation condition*.

A profile implementation condition is true if the referenced profile is implemented.

For example, an Example Fan profile might model fan management. This Example Fan profile might require that the implementation of the *GetAssociatedInstancesWithPath( )* operation for its adaptation of the CIM_Fan class for the detection of instances of the CIM_Sensor representing attached fan speed sensors through the CIM_AssociatedSensor association is conditional on the implementation of an Example Sensors profile for those speed sensors. In this example, an implementation decision is made at the level of implementing the example Sensors profile. The profile implementation condition defined in the Example Fan profile determines the consequences of such profile implementation for the elements adapted in the Example Fan profile.

NOTE There is no restriction that the referenced profile needs to be implemented in the same WBEM service as the referencing profile.

NOTE Implementing a referenced profile for the purpose of conforming to a profile implementation condition in a referencing profile is a design time decision and is not to be confused with detecting profile implementations at run-time. The latter is defined in DSP1033.

7.2.3 Feature implementation condition

A profile may specify a condition based on the implementation of a feature (see 7.12). This kind of condition is called a *feature implementation condition*.

A feature implementation condition is true if the feature is implemented.

For example, an example Fan profile might model fan management. This example Fan profile might define a "FanSpeedSensor" feature. Any elements adapted by the example Fan profile as part of the

FanSpeedSensor feature might be defined as conditional on the implementation of the feature. Likewise, an example Sensors profile modeling the use of sensors might be referenced by the example Fan profile, on the condition that the FanSpeedSensor feature is implemented. In this example, an implementation decision is made at the level of implementing the feature. The two feature implementation conditions defined in the example Fan profile determine the consequences of implementing the feature, in this case the implementation of the elements adapted by the example Fan profile and related to fan speed sensoring, and implementation of the example Sensors profile in the context of fan speed sensors.

NOTE The way this example defines an implementation option in a profile is different from how the example described in 7.2.2 defines that; in this case there is no implementation difference between using a profile implementation condition or a feature implementation condition. However, the use of a feature implementation condition is preferred since it makes explicit a requirement that a set of related elements be implemented as a unit. Additionally, the profile is required to provide a means of detecting that a feature has been implemented. This generally reduces the number of variations in implementations and therefore the complexity of clients that must accommodate those variations.

7.2.4 Class adaptation implementation condition

A profile may specify a condition based on the implementation of a non-mandatory class adaptation (see 7.10). This kind of condition is called a *class adaptation implementation condition*.

NOTE The decision to implement an optional class adaptation - or a conditional class adaptation in the case where the condition is not true - is made by an implementer; consequently requirements related to other elements specified by a profile can be conditioned on the implementation of the class adaptation. A class adaptation implementation condition is not necessarily directly observable by a client; for example, consider the case where no instances of the class adaptation exist.

A class adaptation implementation condition is true if the class adaptation is implemented.

For example, the implementation of fan redundancy might be defined in an Example Fan profile such that the adaptation of the CIM_RedundancyGroup class is defined as optional, and the definitions of any other profile elements related to fan redundancy would then be defined as conditional on the implementation of the adaptation of the CIM_RedundancyGroup class.

NOTE In the example the requirements for some related profile elements are conditioned on the implementation of a class adaptation, in effect causing the related profile elements to be implemented if an according decision is made for the begin of the chain; in this situation the definition of a feature along with respective feature implementation conditions on the class adaptation and the related profile elements is considered a better choice.

Deprecated content - start

7.2.5 Instance existence condition

Instance existence conditions are deprecated; see deprecation notice below.

A profile may specify a condition based on the existence of a particular CIM instance. This kind of condition is called an *instance existence condition*.

An instance existence condition is true if the CIM instance as defined by the profile exists. The profile shall define a discovery mechanism for the CIM instance, for example one or more of the following:

- Define values of the key properties that identify the CIM instance
- Define one or more association paths that connect the CIM instance with CIM instances controlled by the profile or one of its referenced or referencing profiles

For example, a profile that optionally adapts a specialization of the CIM_Service class that has several domain specific service methods might state that the CIM_HostedService association that models the relationship between the service and the system hosting the service shall only be implemented if the CIM_Service instance exists.

NOTE The concept of instance existence conditions is problematic because it implies that the implementation of conditional elements (such as adaptations) depends on the existence of CIM instances. Thus a design time decision (such as implementing a class adaptation) depends on a situation that is the result of an implementation and is observable at runtime only (such as the existence of a CIM instance). As a consequence a profile implementer who needs to make the design time decision (e.g. implement the adaptation) would have to figure out potential runtime situations (e.g. the existence of CIM instances) that are only the result of an implementation; this is considered a cumbersome and potentially error prone exercise.

Deprecation notice: Instance existence conditions are an unnecessary complication and indirection of the decision process for implementing a conditional element. New profiles and revisions of existing profiles should use feature implementation conditions rather than instance existence conditions.

NOTE It is emphasized that the deprecation of instance existence conditions does not prohibit profiles to specify the existence of instances as a means for clients to detect the result of design time decisions. Quite the contrary is the case: This guide encourages the explicit use of the existence of CIM instances to convey the result of implementation decisions. This significantly differs from instance existence conditions insofar as here a design time decision (e.g. the implementation of a class adaptation) is made first, and as a consequence the implementation is required to provide detection elements (e.g. CIM instances) that enable clients to detect the implementation of the conditional element.

Deprecated content - end

Deprecated content - start

7.2.6 Property value condition

Property value conditions are deprecated; see deprecation notice below.

A profile may specify a condition based on the value of a property of a particular CIM instance. This kind of condition is called a *property value condition*.

A property value condition is true if the CIM instance exists and the values of one or more properties in the instance match a pattern defined by the profile.

For example, a profile that adapts a specialization of the CIM_Service class that defines several methods might in addition adapt a specialization of the CIM_Capabilities class that defines an array property and a corresponding value set, where each element of the value set designates one of the methods from the CIM_Service class. Implementation of a particular method would be required if the corresponding value is set as an element of the array property.

NOTE The concept of property value conditions is problematic because it implies that the implementation of conditional elements (such as adaptations) depends on values of properties in CIM instances. Thus a design time decision (such as implementing a class adaptation) depends on a situation that is the result of an implementation and is observable at runtime only (such as a certain value of a property in a CIM instance). As a consequence a profile implementer who needs to make the design time decision (e.g. implement the adaptation) would have to figure out potential runtime situations (e.g. property values in CIM instances) that are only the result of an implementation; this is considered a cumbersome and potentially error prone exercise.

Deprecation notice: Property value conditions are an unnecessary complication and indirection of the decision process for implementing a conditional element. New profiles and revisions of existing profiles should use feature implementation conditions rather than property value conditions.

NOTE It is emphasized that the deprecation of property value conditions does not prohibit profiles to specify property values as a means for clients to detect the result of design time decisions. Quite the contrary is the case: This guide encourages the explicit use of property values to convey the result of implementation decisions. This significantly differs from property value conditions insofar as here a design time decision (e.g. the implementation of a class adaptation) is made first, and as a consequence the implementation is required to provide detection elements (e.g. CIM instances with specific property values) that enable clients to detect the implementation of the conditional element.

Deprecated content - end

7.2.7 Managed environment condition

A profile may specify a condition based on circumstances in the managed environment. This kind of condition is called a *managed environment condition*.

Managed environment conditions are specified in profiles using plain text that refers to the managed environment and its object types.

A managed environment condition is true if the conditions specified in the text are true for the particular type of managed environment for that the profile is implemented.

For example, a profile addressing the management domain of storage host bus adapters might adapt the CIM_FCPort class modeling fiber channel host SCSI initiator ports. The profile might state that the implementation of its adaptations of the CIM_AlarmDevice class and of the CIM_AssociatedAlarm association are conditional, and is only to be implemented if the type of managed environment for that the profile is implemented provides a client callable interface to blink an LED for those fiber channel ports that are represented by instances of the CIM_FCPort class.

NOTE Managed environment conditions allow the formulation of conditions in profiles such that an implementation of the profile is required to implement the conditional element only if respective means are available to the implementation in the particular type of managed environment. In the example above the implementation of the CIM_AlarmDevice class only makes sense if the implementation does have means at its disposal to blink the LEDs.

NOTE Of course managed environment conditions are only testable using white box testing where the test code does also have access to specific means to test the managed environment condition. Ideally these means would be different from those used by a profile implementation.

7.3 Naming conventions for named profile elements

This subclause defines the naming conventions and rules for the names of named profile elements.

The following profile elements are defined as named profile elements:

- profile references (see 7.6.2)
- registry references (see 7.8)
- events (see 7.9)
- features (see 7.12)
- adaptations (see 7.10)

The name shall uniquely identify the element within the scope of a profile.

The name shall conform to the format defined for the ABNF rule IDENTIFIER in Annex A of DSP0004.

The name should be composed of a concatenated sequence of words, with each word starting with a capital letter.

Profile element names are part of the normative definitions of a profile; the rules for backward compatibility and deprecation as defined in 7.14 and 7.16 apply.

For example, StateManagement might name a feature that defines a model for the management of the state of managed objects. Examples of adaptation names are Fan for an adaptation of the CIM_Fan class, or SystemFan for an adaptation of the CIM_SystemDevice association modeling the relationship between systems and fans. Examples of profile reference names are DiskSpeedSensors and DiskTemperatorSensors for two profile reference of an Example Disk profile to an Example Sensors profile for the purpose of modeling disk speed sensors and disk temperature sensors.

7.4 Definition of the profile identification

This subclause defines the elements of a profile identification.

7.4.1 General

A profile shall uniquely identify itself through a registered profile name (see 7.4.2), version (see 7.4.3) and organization (see 7.4.4).

NOTE Profile identification identifies a specific version of a profile, not implementations of a profile. Within one WBEM service there may be multiple implementations of the same profile version.

7.4.2 Registered profile name

The registered profile name should provide end-user recognition and should not include CIM class names.

The registered profile name shall be unique within the defining organization.

The registered profile name shall not be changed in any future version of the profile.

The registered profile name shall not include the word "profile". However, in profile specifications, textual references to other profiles should append the word "profile" to the registered profile name.

NOTE For example, a profile specification would reference a profile whose value of the registered profile name attribute is "System Virtualization" using text such as "If the *System Virtualization* profile (see DSP1042) is implemented, then …".

7.4.3 Registered profile version

The registered profile version shall be the full version of the subject profile. The version shall be defined following the rules for versioning DMTF specifications defined in DSP4004 that requires a version to take the form m.n.u[d[d]], where m is the major version identifier in numeric form, n is the minor version identifier in numeric form, u is the update identifier in numeric form and [d[d]] is the optional draft identifier in alphabetic form. For a registered profile version the major version identifier, the minor version identifier and the update identifier shall always be specified, and a draft identifier shall not be specified.

NOTE This requirement does not preclude profile specification documents to be versioned with a draft identifier, but the registered profile version is never specified with a draft identifier.

7.4.4 Registered organization name

The registered organization name shall be the name of the organization that is publishing the profile. For profiles that are published by DMTF, the registered organization name shall be "DMTF".

7.4.5 Organizational contact

A profile shall identify the organizational unit that is the contact for the profile. For profiles owned by DMTF details are defined in DSP4004.

7.5 Definition of schema references

This subclause defines the elements of a reference to a schema.

7.5.1 General

A profile shall reference each schema that defines classes adapted by the profile. Each schema reference shall state the schema name (see 7.5.3), the schema version (see 7.5.2), and the schema organization (see 7.5.4), unless default values apply.

7.5.2 Schema version

The schema version shall be stated with the major version identifier, the minor version identifier and optionally where needed the update identifier, and shall refer to the earliest version of the schema that meets the requirements of the profile. Regardless of whether or not an update identifier is stated, the

latest published update version with the stated major and minor version identifier is referenced, as defined in DSP4004.

7.5.3 Schema name

The schema name shall refer to the schema by the name that the owning organization assigned to the schema. The specification of this attribute is optional only in the case where only one schema is referenced; if not specified in this case, the default schema name is "CIM".

7.5.4 Schema organization

The schema organization shall refer to the organization that owns the schema. The specification of this attribute is optional only in the case where only one schema is referenced; if not specified in this case, the default schema organization is "DMTF".

7.5.5 Schema experimental flag

Profiles may reference schemas that are designated as experimental by the organization that defines the schema. A reference to an experimental schema shall be marked as experimental.

NOTE See 7.15 for rules for the specification of experimental content.

7.6 Definition of profile relationships

7.6.1 General

A profile (the referencing profile) may define relationships to multiple other profiles (the referenced profiles). A profile may be referenced by multiple other profiles. The effect of referencing is that the definitions of referenced profiles implicitly apply to the referencing profile, without being repeated by the referencing profile. However, the referencing profile may refine definitions and requirements established by the referencing profile.

The definition of profile relationships causes respective implementation requirements; for details, see clause 8 .

A profile shall not reference its previous versions.

The definition of cyclic relationships is allowed; however, there are restrictions for the definition of cyclic relationships between derived profiles. For example, it is not possible to define cyclic relationships between adaptations; for details, see 7.10.2.2 .

An example of a cyclic relationship between profiles is a profile A that defines a mandatory relationship to a profile B, and that profile B defines a mandatory back to profile A. Another example is an autonomous profile that defines a relationship to each of its component profiles, and each component profile refers back to its scoping profile.

NOTE Generally, component profiles do not reference their scoping profile.

7.6.2 Definition of explicit profile relationships

Explicit profile relationships are defined by means of profile references. A profile reference defines a use of the referenced profile within the context of the referencing profile.

It is possible that a particular profile is referenced more than once by a subject profile, where each profile reference would define a separate use of the referenced profile. For example, an Example Fan profile addressing the management domain of fans in systems could reference an Example Sensors profile for the representation of sensors monitoring fan speed, and for temperature sensors monitoring the temperature of cooled elements.

A profile reference is a profile element that references a profile by stating the type of the profile relationship and by identifying the minimally required version of the referenced profile. In addition, the use of the referenced profile in context of the referencing profile should be described.

A profile reference shall be assigned a name following the conventions defined in 7.3 .

The type of the profile relationship shall be either a requirement level, or shall be profile derivation.

The type of explicit profile relationship shall be indicated using one of the following keywords:

- **Mandatory**
  A mandatory profile relationship indicates that the definitions of the referenced profile strictly apply in context of the referencing profile, with no deviation permitted. In this case the referenced profile is termed a mandatory profile of the referencing profile.
- **Derivation**
  A derivation profile relationship indicates that the definitions of the referenced profile are the base for the referencing profile, as detailed in 7.6.3. In this case the referenced profile is called a base profile, and the referencing profile is termed a derived profile. From a client point of view a derived profile is substitutable for a base profile..
- **Conditional**
  A conditional profile relationship indicates that the definitions of the referenced profile under specified conditions apply in context of the referencing profile. In this case the referenced profile is termed a conditional profile of the referencing profile.
- **Optional**
  An optional profile relationship indicates that the definitions of the referenced profile optionally apply in context of the referencing profile, as far as elements affected by these definitions are selected by an implementer. In this case the referenced profile is termed an optional profile of the referencing profile.

NOTE Scoping is an implicit profile relationship; for details, see 7.6.5 .

The identification of the minimally required version of the referenced profile shall be stated with all of the following:

- The registered profile name of the referenced profile (see 7.4.2),
- the major version identifier, the minor version identifier and optionally where needed the update identifier of the registered profile version of the referenced profile (see 7.4.3), and
- the registered organization (see 7.4.4) of the referenced profile.

Regardless of whether or not an update identifier is stated, the latest published update version with the stated major and minor version identifier is referenced.

A referencing profile shall not redefine mandatory definitions of referenced profiles as conditional or optional and shall not redefine conditional definitions of a referenced profile as optional.

A referencing profile shall not duplicate definitions of its referenced profiles unless it establishes additional constraints.

A referencing profile shall not remove any constraints established by its referenced profiles.

If a referenced profile defines an adaptation of a class for a particular purpose, then any adaptation defined by a referencing profile that is based on the adaptation of the referenced profile (see 7.10.2.2) shall address the same or a more specific purpose, and shall adapt the same class or a subclass of the class adapted by the referenced profile. For example, if a referenced profile defines a NetworkPort adaptation of the CIM_NetworkPort class for the representation of physical network connectors, and a referencing profile defines an adaptation based on the referenced profiles NetworkPort adaptation, then the adaptation of the referencing profile shall adapt the CIM_NetworkPort class or a subclass of the CIM_NetworkPort class, but not a more general class such as for example the CIM_LogicalPort class.

7.6.3 Definition of derived profiles

7.6.3.1 General

The rules in 7.6.3 ensure that a client that exploits the management interface defined by a base profile can likewise interact through that management interface with implementations of the base profile or with those of derived profiles.

A derived profile shall be based on exactly one *direct* base profile.

However, profile derivation may be applied at more than one level, such that a base profile likewise may be a derived profile. For example, a profile A may be based on a profile B, and profile B may be based on profile C, and so forth. Consequently a derived profile - while having exactly one *direct* base profile – can have additional *indirect* base profiles.

A derived profile inherits definitions of all its (direct or indirect) base profiles, as follows:

- management domain context
- schema references
- events
- features
- profile references
- registry references
- adaptations (including property requirements, method requirement and operation requirements)
- use-cases

Other definitions of base profiles are not inherited by a derived profile and need to be exclusively defined by the derived profile; in some of these cases definitions of 7.6.3 constrain the possible choices of a derived profile.

NOTE Special implementation requirements apply for derived profiles. For example, all implementation requirements defined by derived profile need to be merged with those of its base profiles; for details, see clause 8 .

Deprecated content - start

Version 1.0 of this guide defined *profile specialization*. This was deprecated and replaced by *profile derivation*, because profile specialization does not address the possible cases of expanding the management domain addressed by, and extending the management interface defined by the base profile.

Deprecated content - end

Deprecated content - start

Version 1.0 of this guide allowed multiple inheritance, such that a derived profile could be directly based on more than one profile. This was deprecated because it enabled the definition of derived profiles while not ensuring polymorphism; i.e., it was not ensured that a client written against the definition of any base profile could interact with the implementation of the derived profile. Furthermore, there were no rules with respect to the merge of implementation requirements resulting from definitions of the base profiles and the derived profiles, and there were no rules that prohibited derived profile from being based on a set of base profiles with contradicting requirements.

Deprecated content - end

7.6.3.2 Propagation of the management domain

A derived profile may address a management domain that may be restricted, expanded or unchanged with respect to the management domains addressed by its (direct or indirect) base profiles. For example if a base profile applies to the management domain of network port management, a derived profile may restrict that to the management of Ethernet network ports.

The management interface defined by base profiles completely become a part of the interface defined by the derived profile for its management domain. This rule ensures that clients exploiting the management interface as defined by a base profile can interact with an implementation of a derived profile to the same extent as with an implementation of the base profile.

A derived profile may define extensions beyond the management interface defined by base profiles.

7.6.3.3 Propagation of constraints

A derived profile inherits constraints on profile elements from its (direct or indirect) base profiles. More specifically, if profile elements defined in base profiles are not redefined in the derived profile, the definitions of the base profiles apply without changes. Also, if a derived profile redefines profile elements defined in its base profiles, the constraints defined in the base profiles apply for the redefined profile elements as stated in the base profiles and without being restated by the derived profile.

A derived profile may specify additional constraints; in this case the additional constraints shall not violate the inherited constraints.

NOTE Implementations of a derived profile are required to satisfies the requirements of all its (direct and indirect) base profiles. Thus a client written against the management interface defined by the base profile works also with an implementation of a derived profile. Implementation requirements are detailed in clause 8.

NOTE The effects of this rule are different with respect to data sent or received by an implementation. For example, if a base profile requires an output parameter to have only the values "4", "5", or "6", definitions in the derived profile are restricted to this value set, but may reduce that to any subset such as for example "4" and "6". However, in the case of an input parameter, the derived profile is not allowed to further reduce the value set, because a client written against the base profile may use all values as defined by the base profile. Consequently, for input/output parameters, a derived profile is unable to modify (extend or reduce) the value set defined by the base profile. Likewise, this applies to properties that are readable and writable.

7.6.3.4 Propagation of requirement levels

A derived profile inherits profile elements with the same requirement level as that defined by its (direct or indirect) base profiles; this means that profile elements defined in base profiles are considered part of a derived profile with the same requirement level, without requiring a new definition in the derived profile.

A derived profile may redefine optional profile elements of its base profiles as conditional or mandatory, and may redefine conditional profile elements of its base profiles as mandatory.

A derived profile may redefine conditional profile elements of its base profiles as conditional. In this case the condition in the derived profile shall be satisfied if the condition in the base profile is satisfied.

NOTE For example, consider a base profile that requires a conditional profile element if either the X feature or the Y feature is implemented, then a derived profile would not be allowed to narrow the condition such that it would require the conditional profile element only if the X feature is implemented. The reason is that a client of the base profile would expect the conditional profile element also be present in the case where the Y feature is implemented.

7.6.3.5 Definition of schema references

A derived profile shall reference each schema that defines classes adapted by the profile and by any (direct or indirect) base profile; see 7.5 for a definition of the elements of schema references.

A derived profile may introduce new schema references.

A derived profile may refine a schema reference of a base profile by requiring a more recent version of the referenced schema.

7.6.3.6 Propagation of the central and scoping class adaptations

The scoping class adaptation of a derived profile shall be based on the scoping class adaptation of its direct base profile. For the adapted class and for other base adaptations the provisions of 7.10.2.2 apply.

The central class adaptation of a derived profile should be based on the central class adaptation of its direct base profile. For the adapted class and for other base adaptations the provisions of 7.10.2.2 apply.

7.6.3.7 Propagation of profile references

A derived profile inherits all profile references defined by its (direct or indirect) base profiles. A derived profile may introduce new profile references. A derived profile may overwrite profile references made in base profiles with profile references that reference profiles derived from the profiles referenced by the base profiles.

Profile references are named profile elements. A profile reference defined in a derived profile intending to overrides a profile reference defined in a base profile should state the name of the profile reference defined in the base profile.

7.6.3.8 Propagation of registry references

A derived profile inherits all registry references defined by its (direct or indirect) base profiles. A derived profile may introduce new registry references.

7.6.3.9 Propagation of features

A derived profile inherits all features defined by its (direct or indirect) base profiles. A derived profile may introduce new features.

If the name of a feature defined by a derived profile is identical to the name of a feature defined in one of its base profiles, the feature defined by the derived profile shall be a refinement of the feature defined in the base profile.

7.6.3.10 Propagation of adaptations

A derived profile inherits adaptations defined by its (direct or indirect) base profiles. There are two cases:

Case A : The derived profile defines a new adaptation that is based on one or more adaptations defined its base profiles, for the same or a more specific purpose. In this case the rules for basing an adaptation on other adaptations as defined in 7.10.2.2 apply. The name of the adaptation defined by the derived profile may differ from the name of the adaptation defined by the base profile.

For example, an Example Ethernet Port profile may define an EthernetPort adaptation of the CIM_EthernetPort class for the representation of Ethernet ports that is based on a NetworkPort adaptation of the CIM_NetworkPort class that is defined by a base Example Network Port profile.

Case B : Adaptations defined by base profiles not referenced as a base adaptation of one of the adaptations defined by the derived profile are propagated without changes into the derived profile, including references to properties, methods and operations. The adaptation name defined by the base profile becomes an adaptation name of the derived profile. If naming conflicts result from this rule, they shall be resolved by the derived profile through the application of case A. For example, naming conflicts may occur in the case where a new release of a base profile defined an adaptation with a name in use by an already existing derived profile.

A derived profile may define new adaptations in addition to those defined by its base profiles.

7.6.3.11 Propagation of use-cases

A derived profile inherits all use-cases defined by its (direct or indirect) base profiles. A derived profile may introduce new use-cases.

A derived profile may extend use-cases defined in base profiles.

7.6.4 Definition of abstract and concrete profiles

7.6.4.1 Abstract profile

An abstract profile is a special kind of profile specifying common elements and behavior as a base for derived profiles. An abstract profile is explicitly designated as abstract. An abstract profile shall not be implemented. An abstract profile may define class adaptations of concrete classes and/or abstract classes.

An abstract profile may be a derived profile, and may be further derived.

Abstract profiles serve two purposes:

- Provide a base for derived profiles
- Provide a point of reference for referencing profiles

For example, an abstract profile could be defined for the management domain of basic computer system management, and derived profiles could tailor that to various types of computer systems such as desktop computer systems or virtual computer systems.

Abstract profiles can also be referenced from concrete profiles. In this case implementations of the management interface defined by the abstract profile shall be provided by implementations of concrete profiles that are derived from the abstract profile.

This concept enables the definition open sets of component profiles. Generally the management interface for a particular management domain is defined by an autonomous profile and a set of referenced component profiles; in this case the autonomous profile and its referenced component profiles form a closed set. However, if the component profiles exhibit a base set of common characteristics, these could be addressed by an abstract profile. The component profiles would be derived from the abstract profile, and the autonomous profile would only reference the abstract profile and optionally a minimum required set of component profiles. Now the set of profiles is open, since it is possible to define additional component profiles based on the abstract profile, without requiring that these are referenced by the autonomous profile, but still being discoverable by clients.

7.6.4.2 Concrete profile

A concrete profile is any profile that is not an abstract profile. Only concrete profiles may be implemented. A concrete profile may be a derived profile, and a derived profile may be based on both concrete profiles and / or abstract profiles.

If a concrete profile defines a class adaptation of an abstract class it shall require that a concrete sub-class of the abstract class is implemented. If possible, the profile should identify to a concrete subclass that shall be implemented in the case where a more specific subclass of the abstract class is not implemented.

For example, if a concrete profile defines a class adaptation of the abstract CIM_Component association then the profile might require the implementation of a concrete subclass of the CIM_Component association, and,might require the implementation of the CIM_ConcreteComponent association in the case where a more specific subclass of CIM_Component is not implemented.

In addition, 7.11 defines requirements for concrete profiles related to profile registration.

7.6.5 Definition of scoping relationships

7.6.5.1 General

Scoping provides a means to subdivide the model defined in context of complex management domains. As detailed in subclauses of 7.6.5, this subdivision is achieved by defining a scoping profile (typically an autonomous profile) that models overall aspects of a management domain, and provides a scope for component profiles (see 7.6.5.3) that each model a subspace of the management domain. Scoping is established by defining a central class adaptation, a scoping class adaptation and a scoping path in each component profile. In addition, scoping provides a means to simplify the profile conformance advertisement of component profile implementations by reducing the number of CIM_ElementConformsToProfile association instances; for details, see DSP1033.

7.6.5.2 Autonomous profile

An autonomous profile defines a management interface for an autonomous and self-contained management domain. An autonomous profile may be defined without relationships to other profiles (standalone), or may be defined with relationships to other profiles that as a set define a management interface for a complete management domain.

7.6.5.3 Component profile

A component profile defines a management interface of a subset or special aspect of a management domain. In most cases it is possible and desirable to specify a component profile independent of its use in context of a particular scoping profile, enabling reuse of the component profile in context of many possible scoping profiles.

For example, an autonomous profile addressing the management domain of systems might reference a component profile for the purpose of addressing the management domain of network ports in system. The same component profile might be referenced by another autonomous profile that addresses the management domain of network switches, in this case for the purpose of addressing the management domain of switch ports.

7.6.5.4 Central class adaptation

The central class adaptation is the focal point of a subject profile. It should model the central managed object type in the management domain that is addressed by the subject profile.

A profile shall designate exactly one mandatory class adaptation as the central class adaptation.

The central class adaptation like any class adaptation, adapts a class defined in a schema. In addition, the central class adaptation shall be based on the CentralElement adaptation defined by DSP1033 (Profile Registration profile), and may be based on additional other class adaptations; for details see 7.10.2.2 .

NOTE The central class adaptation is instrumental in advertising profile implementations, and in identifying profile conformance of profile implementations; for details see DSP1033.

7.6.5.5 Scoping class adaptation

In a component profile the definition of a scoping class adaptation and of a scoping path provide a scope for the central class adaptation.

The definition of the scoping class adaptation in a profile provides an external attach point for use by scoping profiles. A scoping profile may connect to that attach point by defining its central class adaptation based on class adaptations defined in referenced profiles that are designated by those as their scoping class adaptations. As a consequence, the implementation of a scoping profile and its referenced profiles form an agglomeration of profile implementations controlling sets of instances connected through association instances that can be traversed at runtime.

A component profile shall designate exactly one mandatory class adaptation as the scoping class adaptation. That scoping class adaptation shall be different from the its designated central class adaptation (see 7.6.5.4).

An autonomous profile shall either not designate a scoping class adaptation, or shall designate the same class adaptation as both the central class adaptation (see 7.6.5.4) and the scoping class adaptation.

The scoping class adaptation, like any class adaptation, shall be based on a schema class. In addition, the scoping class adaptation shall be based on the ScopingElement adaptation defined by DSP1033 (Profile Registration Profile), and may be based on additional other class adaptations; for details see 7.10.2.2 .

NOTE The scoping class adaptation is instrumental in advertising profile implementations, and in identifying profile conformance of component profile implementations; for details see DSP1033.

7.6.5.6 Scoping path

A scoping path is an association traversal path defined by the subject profile connection its central class adaptation with its scoping class adaptation.

Each component profile shall define a scoping path. The scoping path shall be specified by a set of adaptations of associations and ordinary classes that are defined by the subject profile. The scoping path shall enable bi-directional navigation between instances of the central class adaptation and instances of the scoping class adaptation.

7.6.5.7 Examples of scoping relationships

- Autonomous profile with optional component profiles

Embedded control systems optionally include management interfaces for fans or power supplies. Elements related to core functionality of the control systems defined in the autonomous profile. The fan and power supply elements are defined in separate component profiles.

- Multiple autonomous profiles sharing component profiles

Disk arrays and volume managers provide similar RAID virtualization capabilities from a device of host-resident software. The RAID virtualization component profile is shared by, but mandatory for the Array (external virtualization hardware) and Volume Manager (host-resident virtualization software) autonomous profiles.

- Referenced component profiles, scoped to the same autonomous profile

Many types of systems include batteries – sometimes batteries are configured in redundant sets. This could be modeled as a battery component profile with a separate, optional battery redundancy component profile. Elements of component profiles are scoped to a System instance defined in the context of a top-level autonomous profile in the scoping hierarchy.

- Scoping between component profiles

In some cases, CIM defines scoping between non-system elements. For example, the CIM_ServiceStatisticalInformation class is scoped by the CIM_Service class – which is then scoped to the CIM_System class.

7.7 Definition of the management domain

A profile should define the set of managed object types from the management domain addressed by the profile. These definitions should enable an implementer who implements the profile for a particular type of managed environment to establish a mapping of managed objects from the managed environment to CIM instances that conform to class adaptations defined by the profile.

In some cases it may be sufficient to refer to respective definitions in the schema definition of adapted classes. However, generally profiles adapt a generic classes to model a more specific managed object type that that described in the schema definition the each adapted class.

For example, in Table 1 a simple definition of a management domain by a profile defining a management interface for the management of files and file systems is shown.

Table 1 – Example management domain definition

X-6 Description
This profile addresses the management domain of file management. The major object types are files, directories and file systems. A *file system* is a set of files that are collectively stored. A file system and its files are accessible by clients. Each file system contains one root directory. A *file* is a block of arbitrary information that is stored in a file system. Each file shall have a unique identifier that identifies the file in the scope of a file system. A file may have a name. The name is not required to be unique. The name may be identical with the file identifier. Except the root directory each file is referenced by one or more directories. A *directory* is a special kind of file that contains a list of references to files; each list entry references one file. A directory may assign a name to each referenced file that is unique in scope of the directory.

The management domain definition shown in Table 1 would enable an implementation of the file management profile for the FAT file system to establish a mapping between object types defined by the file management profile and respective elements defined by the specification of the FAT file system.

7.8 Definition of registry references

Profiles may reference message registries and metrics registries.

A registry reference is a profile element that references a registry by stating the type of the referenced registry and by identifying the minimally required version of the referenced registry.

A registry reference shall be assigned a name as defined in 7.3 .

The type of the referenced registry shall be either message registry or metrics registry.

The identification of the minimally required version of the referenced registry shall be stated with all of the following:

- The name of the registry as assigned by the owning organization,
- the major version identifier, the minor version identifier and optionally where needed the update identifier of the registry, and
- the organization that owns the registry.

Profiles may refer to messages defined in message registries, as part of their other definitions.

Profiles may refer to metrics definitions and statistics definitions defined in metrics registries, as part of their other definitions.

7.9 Definition of events

If indications are specified by a profile, then the events that they indicate shall be named and specified by the profile. With the exception of read operations and method invocations (see 6.8), a profile shall define events in terms of the managed environment.

NOTE Lifecycle indications report events in the CIM model, such as the creation, deletion or modification of CIM instances. However, since any change in the CIM model is caused by a previous corresponding change in

the managed environment, this subclause requires profiles to define events in terms of the managed environment.

7.10 Definition of class adaptations

This subclause specifies how profiles define adaptations of classes for a particular purpose.

7.10.1 Purpose of class adaptations

A class adaptation is a named profile element that defines the use of a class defined in a schema for a particular purpose. In addition a class adaptation may be based on one or more other class adaptations. Class adaptations may be referred to simply as *adaptations*.

7.10.2 Requirements for definitions of all kinds of class adaptations

This subclause defines requirements for definitions of all kinds of class adaptations: Adaptations of ordinary classes, adaptations of associations and adaptations of indications.

7.10.2.1 Class adaptation name

A profile shall define a name for each adaptation it defines; the name shall be in conformance with the naming conventions defined in 7.3 .

Deprecated content - start

Profiles that were created in conformance with version 1.0 of this guide in most cases did not define adaptation names, but just stated the name of the adapted class with an optional modifier. Minor revisions of profiles specified in compliance with previous version 1.0 of the guide may continue using the following naming convention for adaptations:

```
AdaptationName = AdaptedClassName [ "(" Modifier ")" ]
```

`AdaptedClassName` is the name of the adapted class. `Modifier` is a short descriptor that describes the use of the adapted class in context of the profile. The modifier should be composed of less than 30 characters.

Examples:

```
CIM_ComputerSystem
CIM_ComputerSystem (Switch)
CIM_StoragePool (Primordial pool)
```

This naming convention shall only be applied for existing definitions of class adaptations in minor revisions of existing profiles. Newly introduced classes in minor revisions shall not apply this naming convention.

Deprecated content - end

7.10.2.2 Adapted class and base adaptations

An adaptation adapts a class defined in a schema for a particular purpose.

In addition, an adaptation may be based on other adaptations, as further defined in this subclause.

For a particular adaptation, the following rules apply:

- **Rule I**: One adapted class.

A adaptation shall identify exactly one class defined in a schema as the adapted class.

- **Rule II**: Zero or more base adaptations.

A adaptation may reference zero or more adaptations defined in the same or in referenced profiles as base adaptations.

- **Rule III**: Compatibility of the adapted class with that of base adaptations.

If a class adaptation A adapts a class C and is based on one or more adaptations A_1 adapting C_1, A_2 adapting C_2, …, A_n adapting C_n, then C shall be the same or a subclass of any C_i, i=1…n .

NOTE The last requirement ensures that an implementation of the subject profile can implement class C without verifying whether a base adaptation requires the implementation of a subclass of C. This enables the supplementary addition of the implementation of a new component profile to a previously existing implementation of a set related profiles, where the new component profile is not referenced by the set of related profiles.

EDITORIAL NOTE: We also considered the following rule:

- If a class adaptation A adapts a class C and is based on one or more class adaptations A_1, A_2, …, A_n, then C and all classes adapted by A_i, i=1…n shall form a single inheritance chain.

In this case the subsequent addition of implementations of "compatible" component profiles to existing implementation sets would require additional provisions by the implementer and not be implicitly ensured by the profile itself.

A class adaptation, its adapted class, its set of base adaptations and their adapted classs form a directed acyclic graph (DAG). This graph is called the span of the class adaptation.

Figure 2 shows an example that illustrates how the rules defined in this subclause establish limitations for the selection of base adaptations or of adaptable classes, once an initial choice is made.

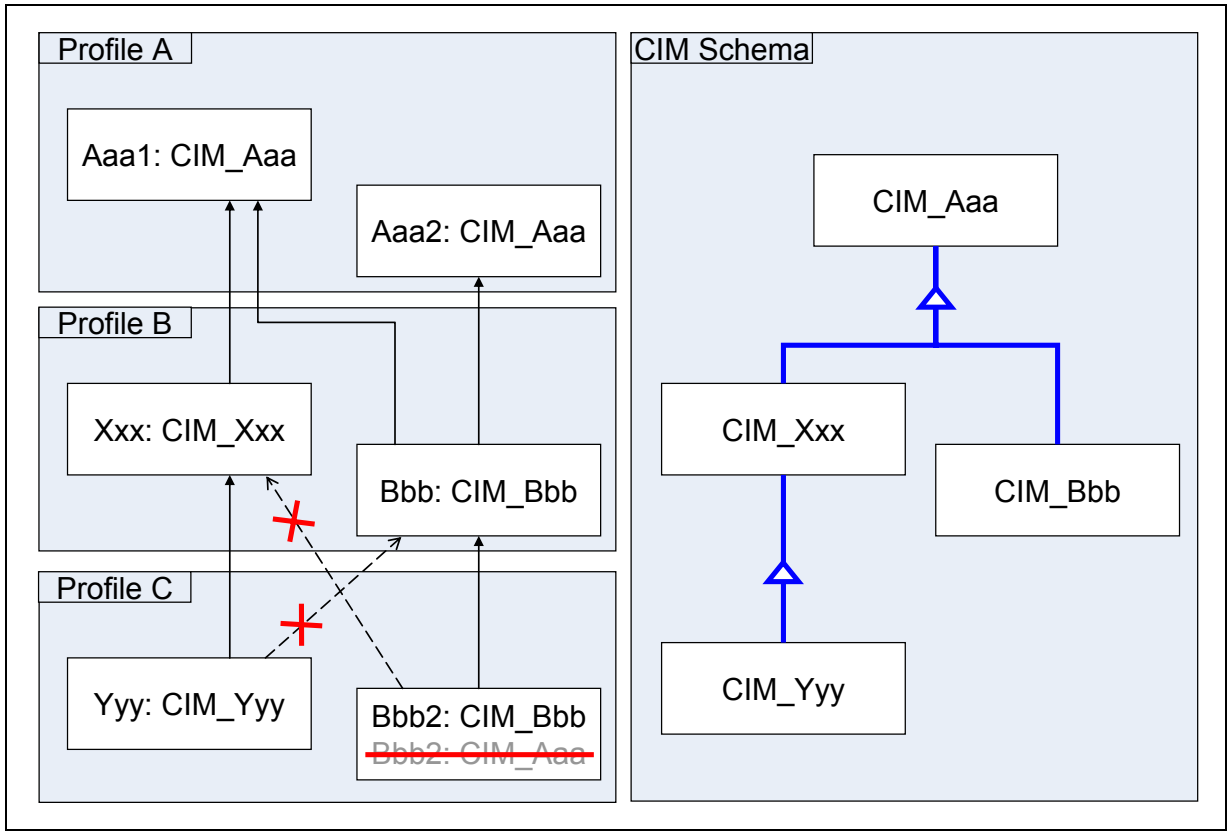


Figure 2 – Class adaptation reference example

In the example shown in Figure 2, the crossed relationships would violate rule II, as follows:

- Adaptation Yyy must not be based on adaptation Bbb because Yyy adapts CIM_Yyy, but Bbb adapts CIM_Bbb that is not CIM_Yyy or a superclass of CIM_Yyy; likewise, adaptation Bbb2 must not be based on adaptation Xxx.
- Adaptation Bbb2 must not adapt CIM_Aaa, because Bbb2 is based on Bbb, and Bbb adapts CIM_Bbb that is a subclass of CIM_Aaa.

Profiles shall not adapt classes that are marked as deprecated in their schema definition, except in the case where a revision of an existing profile retains an adaption of a class that was marked as deprecated in a later version of the schema.

If a class adaptation is based on one or more base adaptations, all of the following rules apply for that class adaptation:

- All definitions and requirements defined by base adaptations are propagated into the new class adaptation.
- The set of instances allowed by the class adaptation shall be a subset of the set of instances allowed by each of its base adaptations.

DMTF collaboration structure diagrams (see 9.2.2.3) are specifically tailored to graphically depict the dependencies introduced by basing class adaptations on other class adaptations.

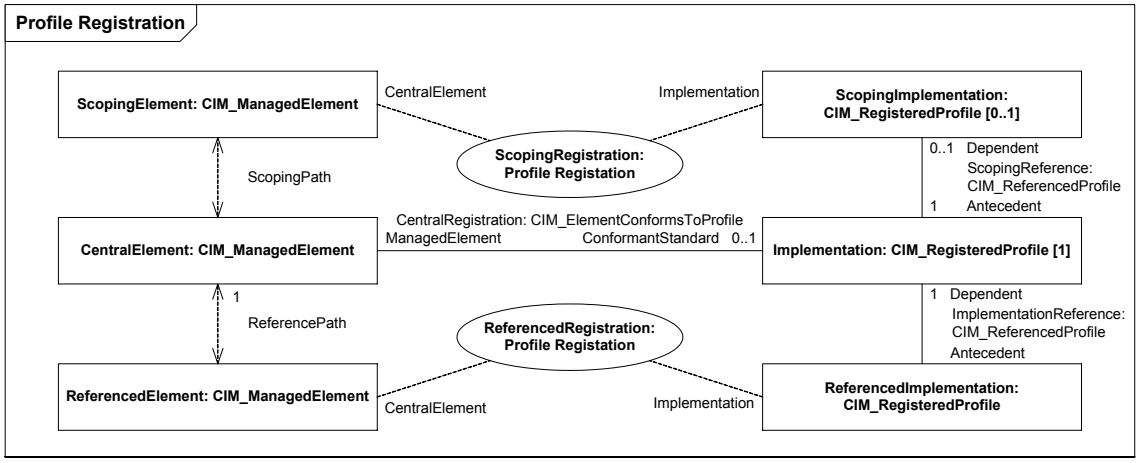


Figure 3 – DMTF collaboration structure diagram of the Profile Registration profile

Figure 3 shows the DMTF collaboration structure diagram of the Profile Registration profile. The CentralElement adaptation (shown as a solid rectangle labeled "CentralElement: CIM_ManagedElement") encapsulates the profile registration related requirements for central elements, the ScopingElement adaptation (shown as a solid rectangle labeled "ScopingElement: CIM_ManagedElement") encapsulates the profile registration related requirements for scoping elements. This guide requires profile to base their central and scoping class adaptations on the respective class adaptations of the Profile Registration profile; see and for further details, see DSP1033.

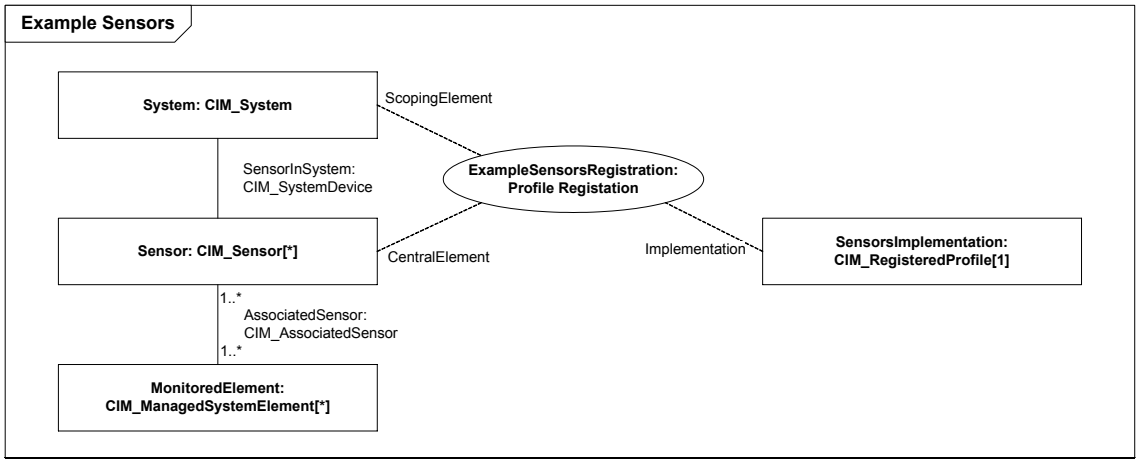


Figure 4 – DMTF collaboration structure diagram of an Example Sensors profile

Figure 4 shows the DMTF collaboration structure diagram of an Example Sensors profile. The dashed oval labeled "SensorsRegistration: Profile Registration" represents the Example Sensor's profile reference to the Profile Registration profile. The solid rectangle labeled "Sensor: CIM_Sensor" represents the Example Sensor's Sensor adaptation of the CIM_Sensor class. The dashed line labeled "CentralElement" indicates that the Sensor adaptation of the Example Sensors profile is based on the CentralElement adaptation of the Profile Registration profile. Likewise, System adaptation of the Example Sensors profile is based on the ScopingElement adaptation of the Profile Registration profile, and the SensorsImplementation adaptation of the Example Sensors profile is based on the Implementation adaptation of the Profile Registration profile.

NOTE An adaptation of the CIM_ElementConformsToProfile association is not shown in Figure 4. A respective adaptation is defined by the Profile Registration profile that addresses all respective requirements; thus there is no need to refine that adaptation in the Example Sensors profile. Also note that the multiplicity of the Implementation adaptation of the Profile Registration profile in Figure 3 is 1, requiring one instance of that adaptation, but that the multiplicity of the ConformantStandard role is 0..1, allowing an implementation to either implement the central class profile advertisement methodology that requires that association for the central class adaptation, or the scoping class profile advertisement methodology defined in the Profile Registration profile that does not require that association for the central class adaptation; for details, see DSP1033.

NOTE The ability of basing class adaptations defined in one profile on class adaptations defined in referenced profiles provides for a much finer granularity of profile dependencies by enabling the introduction of requirements at the level of class adaptations rather than at the level of profiles. For example, the requirement to base the central and scoping class adaptations on respective adaptations of the Profile Registration profile is much stricter than that of version 1.0 of this guide only requiring a reference to the Profile Registration profile.

7.10.2.3 Management domain context

For each class adaptation it defines, the subject profile shall state the managed object type from the management domain that is modeled by the class adaptation. See 7.7 for requirements on defining the management domain and its managed object types.

For adaptations of associations, the management domain context may be specified in the form of a relationship such as for example a containment.

For adaptations of indications, the management domain context may be specified by stating the event that is reported by instances of the adapted indication.

7.10.2.4 Requirement level

For each adaptation it defines, the subject profile shall designate a requirement level. The subject profile may establish further constraints for a adaptation beyond those established by the schema definition of the adapted class, or by referenced adaptations.

7.10.2.5 Definition of property requirements

For each adaptation it defines, the subject profile may define property requirements for properties that are exposed by the adapted class.

Each property requirement shall be designated with a requirement level.

A profile may specify constraints and requirements for as part of property requirements. Any such constraints and requirements apply in addition to, and shall not contradict, any constraints and requirements defined

- in the adapted class and in any of its superclasses,
- in any base adaptation,
- in any class adapted by a base adaptation, and its base classes.

NOTE These rules prohibit any extension of the set of permissible property values as defined by constraints on the adapted class, its base classes, its base adaptations, their adapted classes and their base classes.

As part of every property requirement the profile shall specify a relationship to the aspect of managed objects represented by adaptation instances that is reflected by the property, unless such a relationship is already precisely established by a base adaptation or an adapted class. For example, an Example Fan profile referencing the EnabledState property of the CIM_Fan class in its Fan adaptation would state that the value of the EnabledState property represents the state of the represented fan and relate values of the value set of the EnabledState property to possible fan states.

If string typed properties are referenced, a format as required by 7.10.2.6 should be specified.

A profile may specify a default value for a property. Profile specified default property values apply with regard to output operations in the case where a more specific value is indiscernible by the profile implementation. For example, a profile could define a default value of "" for the (schema required) ElementName property. That value would have to be returned in the case where an implementation is unable to produce a more specific value.

NOTE The semantics of profile defined default values differ from schema defined default values as defined in DSP0004. Schema defined default values only apply with regard to create operations such as for example the CreateInstance( ) operation.

Profiles define constraints as part of property requirements for reference properties in association adaptations, as follows:

- For each reference property a subject profile shall state the adaptation that the reference property refers to. It is required that the referenced adaptation is defined in the subject profile.
- The adaptation referenced by a reference property shall be compatible with the class that is referenced by the reference property in the adapted association class; for details, see 7.10.2.2 .
- Profiles may constrain the multiplicities of references in association adaptations. These multiplicities shall be the same as or narrower than the most narrow multiplicity defined in the adapted class and in any base adaptation.

Profiles shall not define property requirements for properties that are marked as deprecated in the schema definition of the adapted class, except within revisions of existing profiles that retain a property requirement for a property that was marked as deprecated in a subsequent version of the schema after the original version of the profile was released.

7.10.2.6 Formats for string typed properties and parameters

Profiles may specify a mechanism that conveys the format for the values of string typed properties, method parameters and method return values.

For some of the format specification mechanisms that a profile may apply, this guide defines rules that govern the application of these mechanisms, as follows:

- If a profile uses regular expressions to define the format, the regular expressions shall conform to the syntax defined in ANNEX B.
- If a profile uses a grammar to define the format, the grammar shall be stated in ABNF (see RFC5234). A profile may define extensions and modifications to ABNF; if so, these shall be documented in the profile.

NOTE The specification of units is established in schema definitions through the use of the PUNIT or the ISPUNIT qualifiers.

7.10.3 Requirements for definitions of adaptations of ordinary classes and associations

This subclause defines requirements for the definition of adaptations of ordinary classes and for the definition of adaptations of associations.

7.10.3.1 Definition of method requirements

For each class adaptation of ordinary classes or associations it defines, a profile may define method requirements for methods that are exposed by the adapted class.

Each method requirement shall be designated with a requirement level.

A profile may specify constraints and requirements for referenced methods and their parameters. Any such constraints and requirements shall not contradict, and apply in addition, to any constraints and requirements defined

- in the adapted class and its superclasses,
- in any base adaptation,
- in any class adapted by a base adaptation, and its superclasses.

NOTE This rule prohibits any extension of the set of permissible output parameter values and any reduction of the set of permissible input parameter values as defined by constraints on the adapted class, its base classes, its base adaptations, their adapted classes and their base classes.

As part of every method requirement a profile shall specify the method semantics with respect to the managed environment, unless these are already precisely defined by a base adaptation or by the schema definition of an adapted class. The description may adopt text from the schema description of the method, but shall rephrase that as standard English text.

For example, an Example Fan profile referencing the RequestStateChange( ) method of the CIM_Fan class in its Fan adaptation would state that the execution of the method shall effect a state change of the fan that is represented by the Fan adaptation instance on that the method is invoked.

It is generally not sufficient to only describe the expected state of CIM instances after the method execution completes because an implementation is required to effect changes on managed objects in the managed environment, and then reflect respective changes in the CIM instances that represent the managed objects; see 6.7 for further rationale.

If string typed parameters are referenced, a format as required in 7.10.2.6 should be specified.

Profiles shall not define method requirements for methods that are marked as deprecated in the schema definition of the adapted class, except within revisions of existing profiles that retain a method requirement for a method that was marked as deprecated in a subsequent version of the schema after the original version of the profile was released.

7.10.3.2 Definition of operation requirements

For each class adaptation of ordinary classes or associations it defines, a profile shall reference required operations.

Each operation requirement shall be designated with a requirement level.

Operation requirements shall be stated with respect to an operations specification that defines the operations, such as DSP0200 or DSP0223. Profiles based on version 1.0 of this guide referred to DSP0200. Revisions of such profiles conforming to this version of this guide should be updated to refer to DSP0223. References to operations in this guide refer to operations defined in DSP0223 unless otherwise stated.

A profile may specify lists of operation requirements, and refer to these lists when specifying the operation requirements for individual class adaptations.

A profile shall define operation requirements for operations that enable association traversal as part of the operation requirements of association adaptations.

Deprecated content - start

Minor revisions of profiles initially specified in compliance with version 1.0 of this guide may continue defining operation requirements for association traversal operations as part of operation requirements of the adaptations of the classes that are referenced by the adapted association if the association was already defined in the preceding minor version of the profile.

NOTE This rule was changed because in general different operation sets can be defined for any of the associations referencing a particular class. Thus a description of association traversal operation requirements with the referenced class would have to be fanned out into separate operation requirements for each referencing association. Furthermore operation requirements for association traversal operations potentially had to be stated for either association traversal direction. Furthermore, since with this approach the definition of association traversal operation requirements for one association was part of the definition of at least three different adaptations, the verification of completeness of operation requirements for the association adaptation is aggravated.

Deprecated content - end

EDITORIAL NOTE: We are still in the process of determining where to place the association traversal operations: With the association adaptation (as specified here), or with the referenced adaptation(s) as (incompletely) specified by version 1.0 of this guide.

This foilset highlights some of the arguments in either direction:

http://www.dmtf.org/apps/org/workgroup/mrprofiles/download.php/49725/AssociationOperations.ppt

One possible compromise would be to require the specification with the association adaptation, and then refer to that from the referenced class adaptations.

As part of every operation requirement a profile shall specify the operation semantics with respect to the managed environment, unless these are already precisely defined by a base adaptation or by the operations specification.

For write operations (such as for example the ModifyInstance( ) operation defined in DSP0223) it is generally not sufficient to only describe the expected state of CIM instances after the operation execution completes because an implementation is required to effect changes on managed objects in the managed environment, and then reflect respective changes in the CIM instances that represent the affected managed objects; see 6.7 for further rationale.

Profiles may limit the effects of write operations to certain properties. For example, an Example Fan profile that defines operation requirement for the ModifyInstance( ) operation in its Fan adaptation of the

1679 CIM_Fan class may limit the effects of the operation to the EnabledState property, and state that the
1680 operation shall effect a state change of the fan that is represented by Fan adaptation instances.

1681 If a profile requires the ModifyInstance( ) operation to be implemented for a particular class, the default
1682 interpretation is that all non-key properties defined by the class may be modified through the ModifyIn-
1683 stance( ) operation; however, a profile may designate non-key properties as modifiable or unmodifiable.
1684 The operations specification describes the behavior of modification operations with respect to properties
1685 designated as unmodifiable.

1686 Profile shall not define operation requirement for the InvokeMethod( ) operation; this operation is implicitly
1687 required if the profile defines any methods.

1688 Profiles shall not define operation requirement for operations that are marked as deprecated in the
1689 operation specification defining the operation (such as for example DSP0200 or DSP0223), except within
1690 revisions of existing profiles that retain an operation requirement for an operation that was marked as
1691 deprecated in the operation specification after the original version of the profile was released.

EDITORIAL NOTE: We also considered the adding the following rules either in this gude, or in the respective operations specification or in a new profile for operations:

Operations specified in the context of class adaptations carry them a number of implied constraints on named parameters that do not need to be respecified within each profile.

These are:

1) The value of InstancePath shall reference an instance that is a kind of the class of the ClassAdaptation
2) The value of ClassPath shall name a class that is a kind of the class of the Class Adaptation.
3) The value of EnumClassPath shall name a class that is a kind of the class of the Class Adaptation.
4) The value of SourceInstancePath shall reference an instance that is a kind of the class of the Ordinary Class Adaptation named by one of the references of an Association Class Adaptation.
5) The value of AssociationClassName shall be NULL or name a class that is a kind of the class of an Association Class Adaptation. NULL implies the set of all Association Class Adaptations that have at least one reference property constrained to refer to the Ordinary Class Adaptation of the instance named in SourceInstancePath. This set may be further constrained by the values of SourceRoleName and AssociatedRoleName
6) The value of AssociatedClassName shall be NULL or name a class that is a kind of the class of an Ordinary Class Adaptation named by one of the references of the Association Class Adaptation(s) named by AssociationClassName. NULL implies the set of all such Ordinary Class Adaptations. This set may be further constrained by the values of SourceRoleName and AssociatedRoleName.
   If the operation is specified relative to an Ordinary Class Adaptation, then the value of AssociatedClassName name is further constrained to name an opposite referenced class from the point of view of the SourceRoleName and AssociatedRoleName parameters.
7) The value of SourceRoleName shall name a reference property of an Association Class Adapatation belonging to the set named or implied by AssociationClassName. If the operation is specified relative to an Ordinary Class Adaptation, then the named reference is further constrained to refer to the Ordinary Class Adaptation of the instance named in SourceInstancePath.
8) The value of AssociatedRoleName shall name a reference property of an Association Class Adapatation belonging to the set named or implied by AssociationClassName.

Note, that unless further constrained, rules 4-8 enable bi-directional navigation across all associations. While this makes writing a profile simpler, it imposes a greater burden on the implementation to provide

this capability. Profiles that are expected to be widely specialized should specify their interface requirements as narrowly as possible and allow specialized profiles to exend them as needed.

7.10.3.3 Definition of instance requirements

For each adaptation of ordinary classes or associations it defines, a profile shall define instance requirements that precisely define the conditions in the managed environment that require the exposure of adaptation instances namespaces, with the adaptation instances representing managed objects.

7.10.3.4 Concurrency requirements

Each profile should define concurrency requirements with regard to instances of adaptations.

For example, a profile defining requirements for a method or operation may require exclusive access to a subset of the managed environment such that interference from other activities performed on that subset are serialized. However, care should be exercised establishing such requirements, because they might reduce the set of managed environments for that the profile can be implemented.

7.10.3.5 ACID requirements

Profile authors should be aware that protocols, WBEM service infrastructure and adaptation implementations affect the behavior with respect to ACID properties. A profile may define ACID requirements for operations and methods specified by the profile; if specified, ACID requirements shall be defined at the level of the profile defined interface between a WBEM client (or a WBEM listener), and a WBEM service. Profile defined ACID requirements shall be stated in a protocol agnostic way.

NOTE ACID properties for operations and methods are defined in DSP0223. Requirements for ACID properties of operations and methods in some cases are specified by operations specifications (such as for example DSP0223).

If a profile defines operations that create or destroy CIM instances, it shall specify the effects on the managed environment caused by these operations. Profile shall not violate the paradigm that CIM instances at any point in time are required to represent (an aspect of) a managed object.

A profile may specify constraints on the existence of CIM instances with respect to the existence of other CIM instances. This is generally acceptable in situations where these related CIM instance represent different aspects of the same managed object. However, great care should be exercised in the case where dependencies of different managed objects are implied by those constraints.

7.10.4 Requirements for the definition of indication adaptations

7.10.4.1 General

The requirements defined in 7.10.4 for the definition of adaptations of indications apply in addition to the requirements defined in 7.10.2 for the definition of adaptations of all kinds of classes.

7.10.4.2 Indication profile requirements

A profile that defines indications shall reference DSP1054 as a mandatory profile.

7.10.4.3 Indication generation requirements

For each indication adaptation, a profile shall define indication generation requirements. An indication generation requirement shall be defined by stating one or more events, (see 7.8), each of which individually causes the generation of the indication.

In addition, an indication generation requirement may define additional conditions that control the generation of the indication. For example, a profile may state that an indication shall only be generated if the event occurs and listeners are subscribed for that indication.

7.10.4.4 Indication filter instance requirements

Indication filters represent the potential of an implementation to produce a particular indication; for details see DSP1054. Indication filter instance requirements serve the purpose to indicate the fact that a particular indication adaptation is implemented.

For each indication adaptation it defines, a profile shall define an indication filter instance requirement. An indication filter instance requirement shall define a CIM_IndicationFilter instance in terms of its properties such that the instance is bijectively correlatable to the indication adaptation.

There are two approaches for defining indication filter instance requirements:

- The first approach is to define a specific adaptation of the CIM_IndicationFilter class (or a subclass of that) for each indication adaptation. For each of these CIM_IndicationFilter adaptations the profile must define constraints on property values so that only that adaptation is bijectively correlatable to the indication adaptation.
- The second approach is to define a generic adaptation of the CIM_IndicationFilter class (or a subclass of that) that serves for the indication filter instance requirements of more than one indication adaptation. For each indication adaptation it defines the profile must define a specific instance requirement on the generic CIM_IndicationFilter adaptation of the CIM_IndicationFilter with specifically adjusted constraints on property values (overriding or specializing those of the generic CIM_IndicationFilter adaptation) so that only that required instance is bijectively correlatable to the indication adaptation.

DSP1054 defines interdependent instance requirements for CIM_IndicationFilter instances, for CIM_FilterCollection instances and for CIM_MemberOfCollection instances; see DSP1054 for details. A profile may specify additional requirements for the exposure of CIM_IndicationFilter instances in namespaces.

NOTE The requirements defined in this subclause only require a profile that defines indication adaptations to also define related indication filter instance requirements. Indication filter instance requirements only require the definition of CIM_IndicationFilter instances (in terms of properties and values), but not the exposure of such instances in namespaces. However, DSP1054 defines requirements for the exposure of respective CIM_IndicationFilter instances in specific namespaces.

7.10.4.5 Filter collection instance requirements

Filter collections are used to define a collection of indication filters in the context of a particular profile or implementation. Filter collection instance requirements serve the purpose to indicate the fact that one or more logically related indication adaptations are implemented.

For each indication adaptation it defines, a profile may define one or more filter collection instance requirement(s). A filter collection instance requirement shall define a CIM_FilterCollection instance in terms of its properties such that the instance is correlatable to the indication adaptation.

There are two approaches for defining filter collection requirements:

- The first approach is to define a specific adaptation of the CIM_FilterCollection class (or a subclass of that) that defines constraints on property values so that only that adaptation is correlatable to the indication adaptation.
- The second approach is to define a generic adaptation of the CIM_FilterCollection class (or a subclass of that), and define specifically adjusted constraints on property values (overriding or specializing those of the generic CIM_FilterCollection adaptation) so that only that required instance if correlatable to the indication adaptation.

In addition, a filter collection instance requirement shall state the CIM_IndicationFilter instances as required by 7.10.4.4 that shall be considered a member of the filter collection.

DSP1054 defines interdependent instance requirements for CIM_IndicationFilter instances, for CIM_FilterCollection instances and for CIM_MemberOfCollection instances; see DSP1054 for details. A

profile may specify additional requirements for the exposure of CIM_FilterCollection instances in namespaces.

NOTE This subclause only establishes rules for the definition of indication filter collection instance requirements. Indication filter collection instance requirements do not require the exposure of such instances in namespaces. However, DSP1054 defines requirements for the exposure of respective CIM_FilterCollection instances in specific namespaces.

7.10.5 Examples of class adaptations

An example of a simple adaptation that does not establish additional constraints is a profile that addresses the management domain of computer system management, adapts the CIM_ComputerSystem class modeling computer systems and does not specify constraints on properties. A conformant implementation of this profile's adaptation of the CIM_ComputerSystem class is only required to show non-NULL values for the properties defined as mandatory in the schema definition of the CIM_ComputerSystem class (key properties).

Typical examples of adaptations that define additional constraints are:

- A profile addressing the management of systems defining an adaptation of the CIM_ComputerSystem class for the representation of systems, and constraining the set of properties to be implemented to a subset of those defined in the CIM_ComputerSystem class.
- A profile addressing the management of system memory defining an adaptation of the CIM_Memory class for the representation of system memory, and constraining that the value of the EnabledState property shall be 2 (Enabled).
- A profile addressing the management of disks defining an adaptation of the CIM_StorageExtent class for the representation of RAID disks, and constraining that the value of the ErrorMethodology property shall match the pattern "RAID3|RAID4|RAID5".
- A profile addressing the management of floppy disks defining an adaptation of the CIM_DiskDrive class for the representation of floppy disk drives, and constraining that each instance of the CIM_DiskDrive class representing a floppy drive shall be associated with the instance of the CIM_ComputerSystem class representing the containing system.

An example for multiple adaptations of a class in one profile is a profile defining an adaptation of the CIM_AllocationCapabilities class to model the allocation capabilities of a resource pool and to model the mutability of resource allocations.

An example for multiple adaptations of a class in multiple profiles is the CIM_System class that is adapted by many profiles to model very different forms of systems like general purpose systems, network switches, storage arrays or storage controllers. Each of these adaptations is implemented separately, and these implementations need to coexist within one WBEM service.

An example for multiple adaptations of a class in multiple profiles with adaptation dependencies is the adaptation of the CIM_Processor class by two profiles:

- A generic CPU profile defining an adaptation of the CIM_Processor class modeling processors in general. For example, this profile could be implemented for physical processors in physical systems, exploiting management instrumentation provided by software components installed in the physical system. The set of instances controlled by that implementation would be CIM_Processor instances representing host processors.
- A processor resource virtualization profile defining an adaptation of the CIM_Processor class modeling virtual processors, and requiring that this adaptation is based on that of the referenced generic CPU profile. Typically this implies a separate implementation of the referenced generic CPU profile, exploiting management instrumentation provided by the virtualization platform in context of that virtual processors exist. The set of instances provided by that implementation would be CIM_Processor instances representing virtual processors. The advantage resulting from the reuse of the CIM_Processor adaptation is that CIM_Processor instances representing

1826 virtual processors now are visible through the interface defined by the generic CPU profile; con-
1827 sequently, a client could manage the virtual processors through that interface in the same way
1828 as in the physical case. However, it should be noted that in this case the set of CIM_Processor
1829 instances is disjoint from that representing the host processors in the physical case.

1830 As detailed in clause 8, a profile implementation is required to conform to the definitions of the profile and
1831 those of referenced profiles. More specifically, an implementation of an adaptation is required to satisfy all
1832 requirements of all base adaptations, including instance requirements.

1833 7.11 Requirements for profile registration

1834 The CIM schema defines classes that enable the representation of profile implementations and their
1835 relationships, such as the CIM_RegisteredProfile class, and the CIM_ElementConformsToProfile and
1836 CIM_ReferencedProfile associations. DSP1033 (Profile Registration Profile) defines a model for the
1837 representation of profile implementations by defining adaptations of these classes; see DSP1033 for
1838 details.

1839 Concrete profiles shall reference DSP1033 as a mandatory profile; in addition, the requirements of 7.6.5.4
1840 and 7.6.5.5 apply.

1841 7.12 Requirements for the definition of features

1842 7.12.1 Introduction

1843 A feature is a named profile element (see 7.3) that groups the decisions for the implementation of one or
1844 more profile elements into a single decision. This grouping is established by defining the implementation
1845 of other profile elements conditional on the implementation of the feature.

1846 Figure 5 contains an instance diagram for an Example Fan profile, showing CIM instances that represent
1847 a system with two fans, one of which is equipped with a sensor. Instance names are suffixed with the
1848 name of the class adaptation that they conform to (in brackets). The instance diagram also shows the
1849 profile implementations that control respective instances, and how instances of the Fan adaptation
1850 represent different fans in the managed environment.

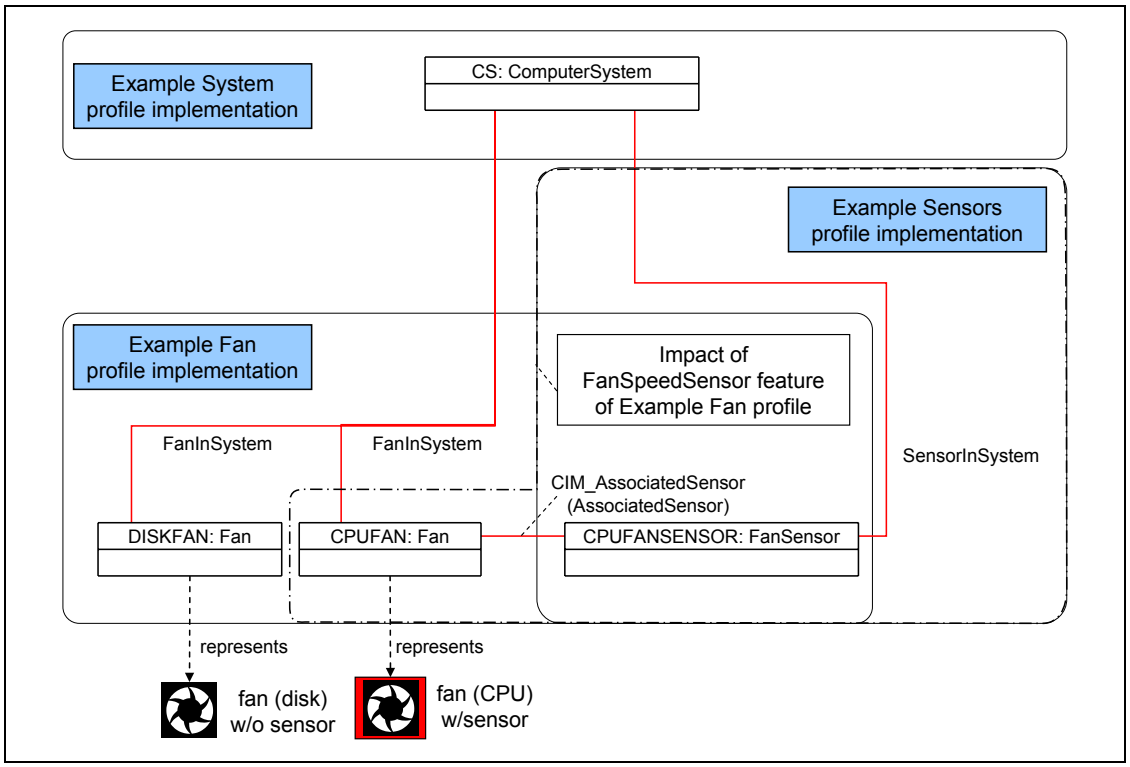


Figure 5 – Instance diagram: Example of FanSpeedSensor feature in an Example Fan profile

In the example shown in Figure 5 an Example Fan profile defines a FanSpeedSensor feature that groups the model elements required to represent and manage the speed of fans. The feature defined functionality impacts several elements defined by the Example Fan profile, such as the representation of the fans themselves, their sensors and their relationship.

In this example it is assumed that the FanSpeedSensor feature is defined as conditional on the existence of fan speed sensors in the managed environment; this is an example of a managed environment condition (see 7.2.7). Consequently an implementer who implements the Example Fan profile for a particular type of managed environment (like for example computer systems produced by a particular vendor) would have to determine whether fans with sensors potentially exist in that type of managed environment. If this is the case then the managed environment condition is true, and the Example Fan profile requires the implementation of the FanSpeedSensor feature.

NOTE It is a typical situation that - as in this example - the implementation of a feature is only required if the managed environment potentially exhibits a particular characteristic (i.e., potentially contains fans with sensors). At implementation time the implementer needs to check whether the characteristic is exhibited by the type of managed environment for that the profile is implemented. If that is the case, then the feature driven implementation requirements become effective and need to be implemented.

The requirements of the FanSpeedSensor feature are defined as conditional requirements with feature implementation conditions (see 7.2.3) on the FanSpeedSensor feature.

In this example it is furthermore assumed that the FanSpeedSensor feature requires the implementation of the Example Sensors profile, and that the Example Fan profile's FanSensor adaptation of the CIM_Sensor class is based on the Sensor adaptation of the Example Sensors profile. This implicates the requirements of the Example Sensors profile for its Sensor adaption into those of the Example Fan profile for its FanSensor adaptation, including the Example Sensor profile's requirements to implement the AssociatedSensor association adaptation and the SensorInSystem association adaptation.

Another assumption is that the example Fan profile defines the FanSpeedSensor feature with a granularity of "Fan instance", and defines the preferred discovery mechanism for the feature by stating that the

feature is supported for a particular Fan instance if a FanSensor instance is associated. This in effect requires the feature implementation to provide feature required elements only for those Fan instances that represent a fan with a sensor.

NOTE Features with instance granularity allow to mandate the exposure of respective instances (i.e., the representation of a fan sensor by a FanSensor instance) only in those cases where the characteristic applies to a particular represented managed object (e.g. the fan has a sensor). Feature implementations need to detect and respectively handle these situations at runtime. Typically feature discovery for features with instance granularity is also defined on a per instance basis, such that from a client perspective the feature is present only for instances exposing the characteristic.

In this example a client would discover the presence of the FanSpeedSensor feature for a particular Fan instance by traversing the CIM_AssociatedSensor association from the Fan instance to the CIM_Sensor instances representing sensors measuring the fan. If such instances exist, they represent respective sensors, and conform to all requirements as defined by the FanSpeedSensor feature.

7.12.2 General feature requirements

A feature should bear a relationship to a functionality in the profile or in the management domain. Profiles shall provide a functional description of each defined feature.

Profile should preferably define a feature instead of a chain of interdependent definitions in order to make decision points more explicit for implementers and ease the discovery of implementation capabilities for clients.

7.12.3 Feature name

A profile shall define a name for each feature it defines; the name shall be in conformance with the naming conventions defined in 7.3 .

7.12.4 Feature requirement level

Profiles shall define their own features with a requirement level of optional or conditional.

Profiles may define constraints on the implementation of features defined within the same or within referenced profiles; for example, a referencing profile may require implementation of a feature that is defined as optional in a referenced profile.

7.12.5 Feature granularity

Feature granularity affects the discoverabiliy and availability of features. Two kinds of feature granularity are possible: Profile granularity and instance granularity.

- Features with profile granularity are either generally available or not available. Feature discoverability is defined at a global level, such that if the feature is available, it is available for all instances affected by feature definitions.
- Features with instance granularity are available only for certain instances. Feature discoverability is defined at an instance level, such that the availability of the feature is indicated only for certain instances.

Profile shall define the granularity of each feature by indicating whether the feature is defined with either profile granularity or with instance granularity.

An example of a feature with profile granularity might be a FanStateManagement feature of an Example Fan profile. If the feature is available (and discoverable for example by means of a property value in a global capabilities instance), fan state management is available for any instance of that profiles Fan adaptation.

In another example (detailed 7.12.1) a FanSpeedSensor feature might be defined with a granularity of "Fan instance" and conditioned (with a managed environment condition) to be implemented only if the managed environment contains fans with sensors. In this case the implementation of the feature would

provide – and a client would be able to discover - feature defined functionality only for those instances of the Fan adaptation that represent fans with sensors, while other instances of the Fan adaptation would not be affected by the feature implementation, and the presence of the feature could not be discovered through those instances.

7.12.6 Feature discovery

Clients need to be able to discover feature implementations.

A profile shall define one and may define more than one mechanism that facilitates discovery of a feature implementation as part of a profile implementation. Each discovery mechanism shall be defined such that the mechanism indicates presence of the feature implementation only if the feature is implemented.

If more than one feature discovery mechanism is defined, one of them shall be designated as preferred.

An example of a discovery mechanism is the presence of a global instance such as for example a CIM_RegisteredProfile instance that represents the implementation of a referenced profile by being associated via the CIM_ReferencedProfile association to the CIM_RegisteredProfile instance representing the implementation of the subject profile.

Another example of a feature discovery mechanism is a specific value constraint for a property value in a global capabilities instance. For example, an Example Fan profile could define the preferred discovery path for the implementation of its FanElementNameEdit feature by requiring that if the FanElementNameEdit feature is implemented there is one globally accessible instance of the CIM_EnabledLogicalElementCapabilities class for that the value of the ElementNameEdit property is true. Global access could be achieved for example by associating that instance to the CIM_RegisteredProfile instance representing the Example Fan profile implementation, or by requiring each Fan instance being associated to the global CIM_EnabledLogicalElement instance through the CIM_ElementCapabilities association. The latter approach is less general insofar as in this case the feature implementation is discoverable only in the case where fans exist in the managed environment.

The discovery mechanism in the previous paragraph could be modified for features with instance granularity by requiring specific capabilities instances instead of global ones.

Another example of a discovery mechanism applicable for features with instance granularity is the presence of an associated instance in context of an instance for that the feature can apply. For example, this is the case for the Fan instances described in the example in 7.12.1, where only in the case where the FanSpeedSensor feature is supported for those fans that are represented by Fan instances with an associated FanSpeedSensor instance.

7.12.7 Feature requirements

Feature requirements define the consequences of implementing a feature. A profile shall define feature requirements in terms of requiring otherwise optional profile elements as conditional with feature implementation conditions (see 7.2.3), or by defining additional constraints. Profiles shall use the following mechanisms to define feature requirements:

- Defining profile elements as conditional with respect to the feature implementation; this applies to
  - profile references
  - otherwise optional or conditional elements within referenced profiles, such as features, class adaptations, properties or methods.
  - adaptations
  - property requirements in adaptations
  - method requirements in adaptations
  - operation requirements in adaptation

- Defining constraints that depend on implementation of the feature

NOTE Clause 8 defines requirements for implementations of profiles, including those of conditional profile elements. See clause 8 for the implementation requirements resulting from features.

7.12.8 Feature example

Figure 6 shows two DMTF collaboration structure diagrams that detail the collaboration defined by an Example Fan profile. For respective diagrams of the Profile Registration profile (referenced in both parts of Figure 6) and an Example Sensors profile (referenced in the lower part of Figure 6), see 7.10.2.2. For details on composite structure diagrams (CSDs), see 9.2.2.3.

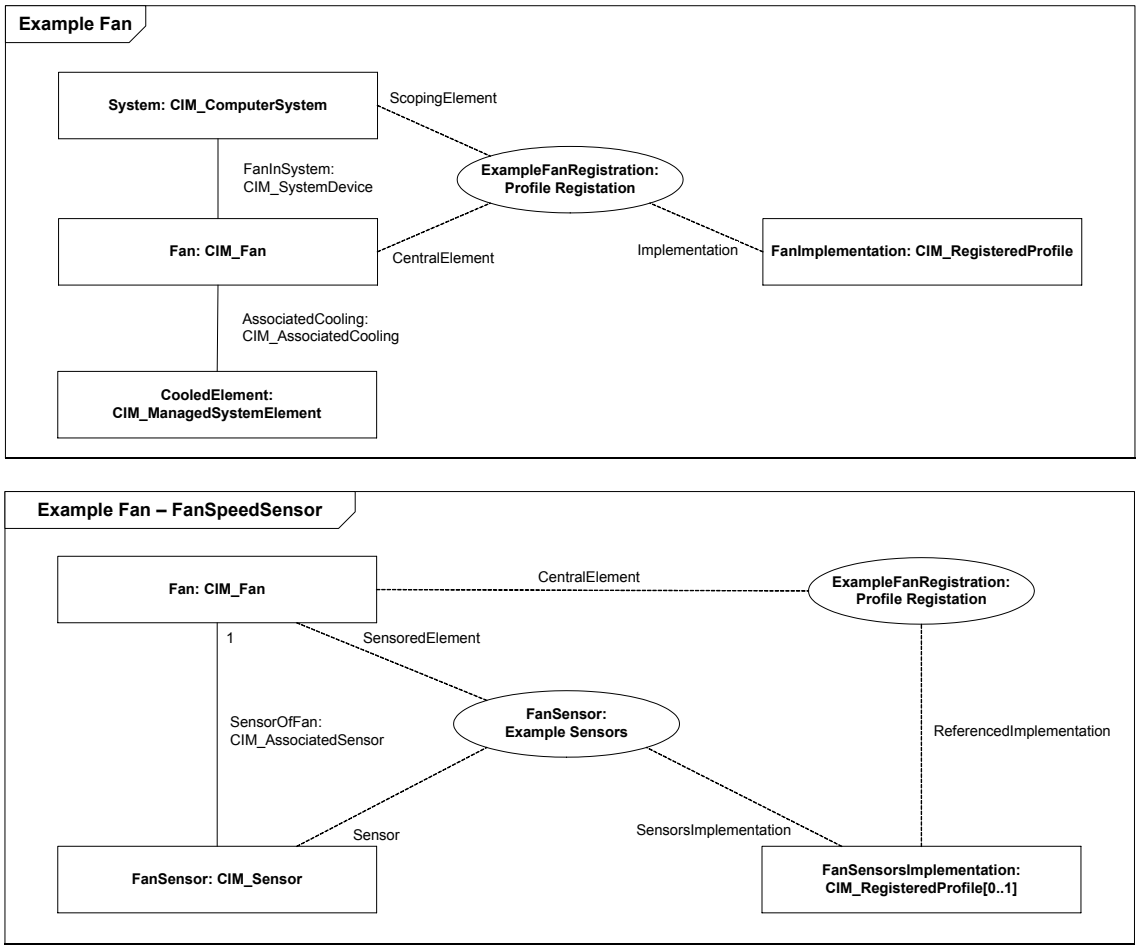


Figure 6 – Examples of DMTF collaboration structure diagrams

The upper diagram in Figure 6 depicts the mandatory class adaptations defined by the Example Fan profile, and how adaptations of the Example Fan profile are based on the adaptations defined in the Profile Registration profile. It also shows implied instance requirements: For example, the Fan adaptation is based on the CIM_Fan class as indicated by the class name that follows the colon. The implied multiplicity [*] of the Fan adaptation indicates that zero or more instances are required to exist at any time. The association end multiplicity of 1 shown at the upper end of the SensorOfFan association adaptation in the lower diagram of Figure 6 indicates that each fan sensor provides sensor information of exactly one fan.

The lower diagram in Figure 6 depicts the class adaptations of the Example Fan profile that contain requirements of its FanSpeedSensor feature. For example, the Example Fan profile defines a relationship to the Example Sensors profile, as depicted by the FanSensorsImplementation adaptation on the right

side with a multiplicity of [0..1]; this means that there are definitions in the Example Fan profile that under certain conditions rely on definitions in the Example Sensors profile.

In this example, it is assumed that the Example Fan profile defines a FanSpeedSensor feature that is conditional on the existence of fans with sensors in the managed environment, and further that individual fans in the managed environment may or may not have sensors. However, these conditions cannot be expressed in the CSD, and in any case need to be stated in the form of normative definitions in the Example Fan profile.

In this example it is further assumed that the primary discovery path for the FanSpeedSensor feature is defined such that the feature is present if a FanSensorsImplementation instance is associated through the CIM_ReferencedProfile association to the FanImplementation instance representing the profile implementation. This is depicted in the lower part of Figure 6 on the right side by showing the FanSensorsImplementation adaptation of the Example Fan profile based on the ReferencedImplementation adaptation of the Profile Registration profile that in turn requires the implementation of the CIM_ReferencedProfile association to the CentralElement adaptation. Thus a client inspecting an implementation of the Example Fan profile as represented by a FanImplementation instance can detect that the FanSpeedSensor feature is implemented by traversing the CIM_ReferencedProfile association to a FanSensorsImplementation instance. If that instance exists, then the FanSpeedSensor feature is implemented in general; however, since in this example the FanSpeedSensor feature is defined with a granularity of "Fan instance", the feature supports only those Fan instances that represent fans with sensors.

If the FanSpeedSensor feature is implemented, then all other profile definitions that are conditioned by a respective feature implementation condition effectively become mandatory; see clause 8 for an algorithm allowing the determination of all implementation-required profile elements in context of the implementation of one or more related profiles. Particularly in this example, each fan equipped with a fan speed sensor needs to be represented by a Fan instance that is based on the MonitoredElement adaptation of the Example Sensors profile.

7.13 Requirements for the definitions of use-cases

7.13.1 General

Profiles should define use-cases that demonstrate the use of the interface defined by the profile. The purpose of use-cases is to illustrate the steps required to accomplish some goal and the effects on managed objects in a managed environment and their CIM representation in the course of accomplishing that goal.

A use-case defines the interaction of an external client and an implementation in the execution of steps required to be performed in the realization of functionality defined in the profile. Clients may be programs like for example CIM clients or other external entities like for example a person using a switch attached to the system. Use-cases should represent a complete task from the perspective of the client; this may involve multiple CIM operations or methods.

It is emphasized that use-cases do not define functionality. Instead, use-cases *apply* functionality that is defined by the profile. For that reason use-cases are not considered as normative elements of a profile, but as essential informative parts that detail potential client activities enabled through implementations of the profile.

NOTE The definition of use-cases given in this subclause calls for a precise formal specification of the invocation of methods and operations that are fully specified by the profile and its referenced specifications. This definition of use-cases is different from that commonly used in software development where a use-case informally describes a required behavior of a yet to be developed software component.

Use-cases should not contain or repeat normative requirements. Normative requirements are defined by other part of the profile such as the definition of class adaptations. However, the description use-cases may informally detail expected effects in the managed environment and respective changes in the CIM model defined by the profile.

Each required operation or method should be included in at least one use-case. A use-case may apply zero or more methods, and a particular operation or method may be applied by more than one use-case.

7.13.2 Requirements for the definition of preconditions

For each use-case the preconditions shall be defined.

Preconditions describe the initial state of an instance of the CIM model defined by the profile. Preconditions should be stated in terms of CIM instances, their property values and the managed object represented by them. Only CIM instances that are involved in the processing of the use-case need to be described. In exceptional cases preconditions may be stated in terms of the managed objects only.

Preconditions may refer to the outcome of other usecases, enabling chaining of use-cases.

7.13.3 Requirements for the definition of flows of activities

Flows of activities should be stated as sequences of steps; however, steps may be skipped or iterated depending on the result of other steps.

Each step should be described in terms of methods and operations that are defined by subject profile or by referenced profiles in the form of method requirementadaptations.

For each use-case step the following types of should be stated:

- the instance on which an operation or method is performed
- the name of the operation or method
- the names and values of all input parameters that are required or accepted
- the expected effect on the managed environment
- the corresponding changes on the CIM model
- the names and values of all output parameters
- the expected return codes, and the corresponding situations that result in the managed environment
- the expected exceptions, and the corresponding situations that result in the managed environment

Use-cases may refer to other use-cases, such that the steps defined by the referenced use-cases are effectively embedded as part of the referencing use-case.

7.13.4 Requirements for the definition of postconditions

For each use-case the postconditions shall be defined.

Postconditions describe the state of the CIM model defined by the profile after the use-case was processed. Postconditions shall be separately defined for the various possible outcomes of processing the use-case, such as success and failures. Postconditions should be stated in terms of CIM instances, their property values. Postconditions may by stated in terms of managed objects. In exceptional cases postconditions may be stated in terms of the managed objects only.

NOTE Note that as described in 6.7 the effect of executing a method or operation on a CIM instance first effects a change in the managed object in the managed environment that is represented by that CIM instance; only after that change is processed, the CIM instances representing aspects of the changed managed object will exhibit corresponding changes in terms of changed property values. However, the state of managed objects may change fast and frequently; consequently it is possible that the state of a managed object as viewed through a CIM instance obtained by a client in a subsequent step after the execution of a use-case exposes a state that already differs from the state that is expected as the result of the use-case execution.

7.14 Backward compatibility

This subclause defines rules for maintaining backward compatibility between versions. Backward compatibility is a characteristic of profiles enabling clients written against a particular minor version of a profile to use the functionality specified by that version in context of an implementation of a later minor version of the profile, without requiring modifications of the client.

Backward compatibility relates to the set of minor versions of the profile with the same major version number. A specific version of a profile shall be backward compatible to its previous minor versions. For example, the version 2.4 of a profile shall be backward compatible to versions 2.0, 2.1, 2.2, and 2.3. A new minor version may extend the functionality of previous versions.

A change that breaks backward compatibility is termed incompatibility.

Incompatibilities may be introduced in new major versions.

Incompatibilities should not be introduced in new minor versions or in new update versions. However, if incompatibilities are introduced in new minor versions or in new update versions, each incompatibility shall be described from a client perspective, and shall state both the version it breaks, and the version introducing the incompatibility.

7.15 Definition of experimental content

A profile may designate definitions as experimental.

A profile that uses experimental schema elements shall designate the definitions that use the experimental schema elements as experimental.

7.16 Deprecation of profile content

A new minor or update version of a profile may deprecate the definition of profile elements or other profile definitions. All deprecated profile definitions shall be continuously documented in new minor or update versions of a profile.

Deprecated profile definitions may be removed in new major versions of the profile.

Profiles should not use deprecated profile content (from other profiles) or deprecated schema elements.

8 Profile implementation requirements

This clause defines the requirements for implementation of one or more profiles. The intended audience targeted by this clause are particularly implementers of profiles.

8.1 Merging implementation requirements from one or more profiles

8.1.1 Motivation

Consider a set of component profiles that each define adaptations of the CIM_ComputerSystem class as their scoping class adaptation, and an autonomous profile that defines its central class adaptation as an adaptation of the CIM_ComputerSystem class and bases that on all the scoping class adaptation adaptations defined by the component profiles. If this set of profiles is implemented, all the requirements from the separate class adaptations of the CIM_ComputerSystem class in the component profiles, and that of the CIM_ComputerSystem class in the autonomous profile would have to be merged into an effective class adaptation of the CIM_ComputerSystem class, resulting in an effective set of requirements such that the implementation conforms to the requirements of all separate class adaptations.

Figure 7 shows an example with three profiles: An Example Fan profile, and Example Profile Registration profile and an Example Sensors profile. The notational conventions applied in Figure 7 deviate from that of DMTF class diagrams in order to convey information that is essential for the merge process. For example, selected properties and constraints on properties are shown in order to demonstrate how

property requirements are merged and lastly have an impact on property values in CIM instances. Also the names of some associations are abbreviated in order to fit the content into one diagram.

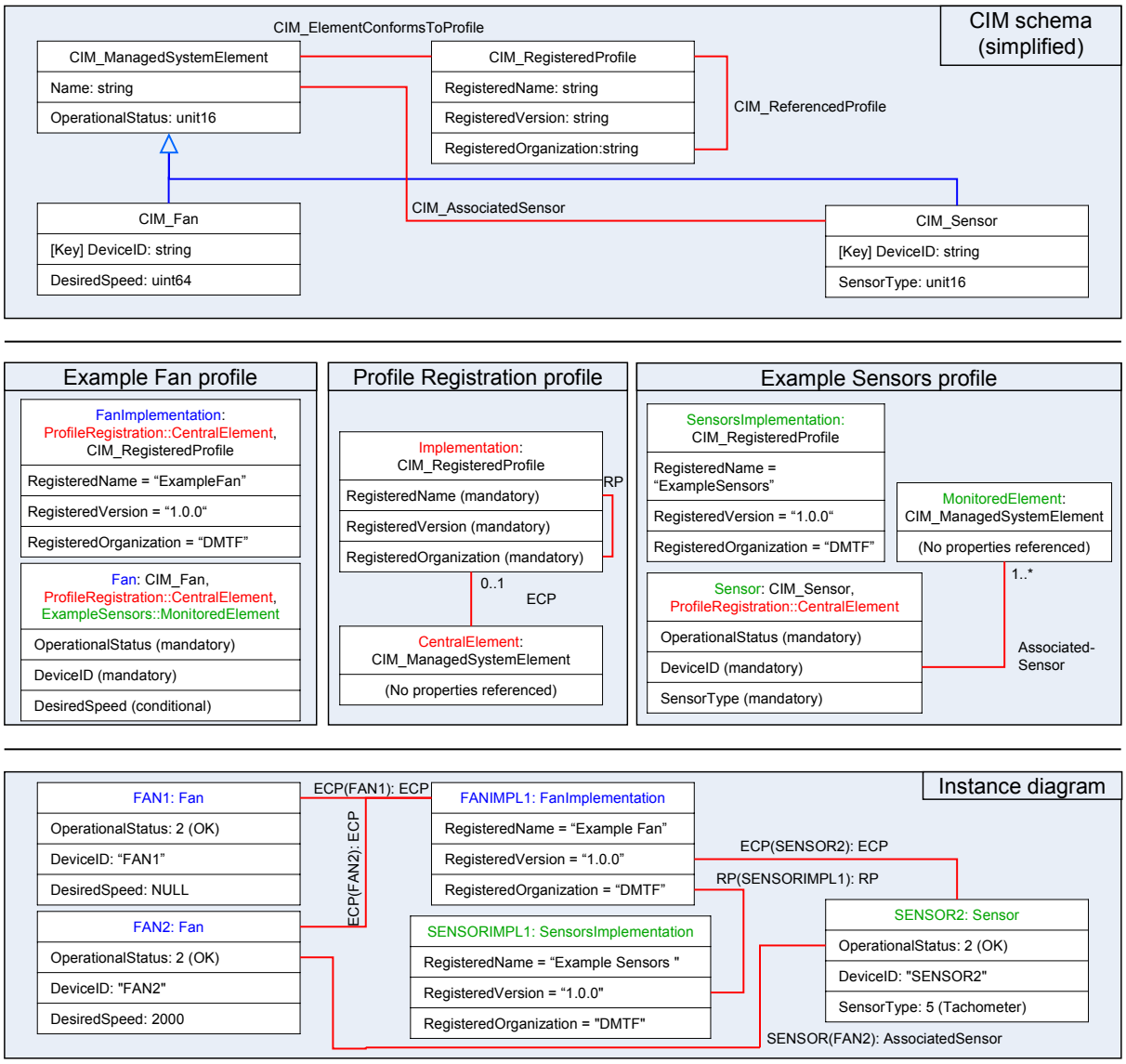


Figure 7 – Class adaptations and instance requirements

The upper part in Figure 7 depicts definitions from the CIM schema.

The center part in Figure 7 depicts selected class adaptations and association adaptations defined by the Example Fan profile, the example Profile Registration profile and the example Sensors profile. Dependencies to schema definitions and to other adaptations are shown as dash-dotted arrows.

The lower part in Figure 7 shows an instance diagram with instances that conform to these adaptations. The topmost box of each instance shows the name of the instance, followed by the class name and the names of one or more adaptations that the instance conforms to.

The Fan adaptation of the Example Fan profile adapts the CIM_Fan class and is based on the CentralElement adaptation of the Profile Registration profile and on the MonitoredElement adaptation of the Example Sensors profile. Consequently, an implementation of the Fan adaptation would have to conform to the requirements of the Fan adaptation itself, that of the CIM_Fan class, that of the CentralElement adaptation defined by the Profile Registration profile and that of the MonitoredElement adaptation defined

by the Example Sensors profile. The latter requirement implies that Fan instances representing fans with sensor are associated with the Sensor instance representing their sensor through an instance of the AssociatedSensor association adaptation.

In Figure 7 it is assumed that the managed environment contain two fans, one without and one with a sensor. Consequently, the implementation provides two Fan instances: FAN1 represents the fan without sensor and FAN2 represents the fan with sensor. Both instances conform to the Fan adaptation defined by the Example Fan profile, and also conform to the CentralElement adaptation defined by the example Profile Registration profile. In addition, FAN2 also conforms to the SensoredElement adaptation defined by the example Fan profile.

8.1.2 Determination of effective implementation requirements

Typically a profile is not implemented by itself, but as part of a set of related profiles that together define a more comprehensive management interface for a management domain. In this case the requirements of all profiles in the set need to be combined. The determination of whether the implementation of a particular profile or profile element is required depends on implementation decisions, on profile definitions and on conditions in the managed environment for that the profiles are implemented.

This subclause details a sequence of activities that yields the set of profiles and the set of adaptations required to be implemented, and merges the requirements from all involved profiles, specifications and schemas.

The implementation profile set is defined as the set of profiles that is implementation-required.

The implementation adaptation set is defined as the set of adaptations required to be implemented in context of an implementation profile set. The process for the determination of the implementation adaptation set described in this subclause first adds all required or selected class adaptations to the implementation adaptation set, and in a subsequent step adjusts that for the effects of basing class adaptations on other class adaptations. After completing the process the implementation adaptation set contains only implementation-required effective class adaptations.

The implementation profile set and the implementation adaptation set are determined as follows:

1) Select an initial desired set of profiles to be implemented, and add these to the implementation profile set.

   NOTE This selection is made by the implementer, and is ruled by the management functionality an implementation intends to provide to clients.

2) For all profiles most recently added to the implementation profile set (either in step 1) before the first iteration of step 2), or otherwise in the previous iteration of step 2)), add their related profiles to the implementation profile set, as follows:

   a) Add all mandatory profiles of the current profile and all its (direct or indirect) base profiles to the implementation profile set, not creating duplicates.

   b) Decide which optional profiles of the current profile are to be implemented, and add these to the implementation profile set, not creating duplicates.

   c) Check the condition of any conditional profile of the current profile and all its (direct or indirect) base profiles. If optional or conditional profile elements affect the evaluation of the condition, decide or determine whether these profile elements are implemented. If the result is yes, perform the following steps for all profile elements contributing to the resolution of the condition:

      - For a class adaptation, add the class adaptation, to the implementation adaptation set, not creating duplicates.
      - For a property, method or operation, add their referencing adaptation to the implementation adaptation set (not creating duplicates), and mark the property / method / operation as implementation-required.

If a condition is met, add the conditional profile to the implementation profile set, not creating duplicates.

3) If any profiles were added to the implementation profile set in the last iteration of step 2), repeat step 2) for the added profiles.

At this point the implementation profile set has been determined. However, the implementation adaptation set contains only those adaptations that influenced the determination of conditions.

4) For all profiles in the implementation profile set determine the set of implementation-required adaptations, as follows:

   a) Add all mandatory adaptations of the current profile and all its (direct or indirect) base profiles to the implementation adaptation set, not creating duplicates. Adaptations of ordinary classes that are referenced by a mandatory association adaptation need to be added as well.

   b) Decide which optional adaptations of the current profile and all its (direct or indirect) base profiles are to be implemented, and add these to the implementation adaptation set, not creating duplicates. Adaptations of ordinary classes that are referenced by an optional association adaptation selected for implementation need to be added as well.

   c) Check the condition of any conditional adaptation of the current profile and all its (direct or indirect) base profiles. If optional or conditional properties, methods or operations affect the evaluation of the condition, decide or determine whether or not to implement these elements. If the decision is yes, add the adaptation referencing the property, method or operation to the implementation adaptation set (not creating duplicates), and mark the property, method or operation as implementation-required. Adaptations of ordinary classes that are referenced by a conditional association adaptation with the condition resolving to true need to be added as well.

At this point, the implementation adaptation set is complete. However, it needs to be adjusted for the effects of basing adaptations on other adaptations. This process shrinks the implementation adaptation set by first determining effective class adaptations, and then merging the requirements from their base adaptations into the effective class adaptations, such that subsequently all base adaptations that are not effective class adaptations can be removed from the implementation adaptation set.

5) Determine the effective class adaptations in the implementation adaptation set. Effective implementation adaptations are those adaptations that have unique implementation requirements that are not completely addressed by a sub-adaptation. First of all, it is evident that all leaf adaptations are effective class adaptation. In addition, there may be adaptations that while also being a base adaptation for some other effective class adaptation need to be implemented separately because of different implementation requirements.

6) For each effective class adaptation, collect all requirements from all its direct or indirect base adaptations and merge them into the effective class adaptations, as follows:

   a) Mark any property, method or operation that is implementation required in any base adaptation as implementation-required in the effective class adaptation.

   b) Merge all implementation requirements from all base adaptations into the implementation requirements of the effective class adaptation, as follows:

      - For properties, collect all defined value constraints and perform a "logical and" operation between them.

      - For methods and operations, merge the semantic requirements, collect all defined value constraints for parameters and perform a "logical and" operation between them.

      - For instance requirements, collect all the instance requirements and perform a "logical and" operation between them.

7) Drop all base adaptation that are not theirselves effective class adaptations from the effective class adaptation set.

At this point, the implementation adaptation set is complete and adjusted for the effects of basing class adaptations on other class adaptations. Next the requirements from the schema need to be merged into the effective class adaptations.

8) For each effective class adaptation, perform the following activities:

   a) Mark each property that has an effective Key qualifier value of true in the schema definition of the adapted class as implementation-required.

   b) Mark each property that has an effective Required qualifier value of true in the adapted class as implementation-required.

   c) Merge all implementation requirements from the adapted class into the implementation requirements of the effective class adaptation, as follows:

      – For properties, collect all defined value constraints and perform a "logical and" operation between them.

        NOTE Schema implied constraints may be induced by means of the ClassConstraint and PropertyConstraint qualifiers.

      – For methods and operations, merge the semantic requirements, collect all defined value constraints for parameters and perform a "logical and" operation between them.

      – For instance requirements, collect all the instance requirements and perform a "logical and" operation between them.

        NOTE Schema implied instance requirements on adaptations referenced by associations may be induced by means of the Min and Max qualifiers in referencing associations.

At this point, each adaptation in the implementation adaptation set defines its effective set of implementation requirements.

8.1.3 Implementation of deprecated definitions

Implementations shall conform to profile definitions regardless of whether they a deprecated or not. Clients should not rely on or exploit deprecated profile definitions, and are encouraged to stop exploiting deprecated functionality as soon as possible.

8.1.4 Example for the determination of effective profile requirements

Figure 8 details an example that is used to describe the algorithm introduced in 8.1.2 for the determination of effective profile requirements.

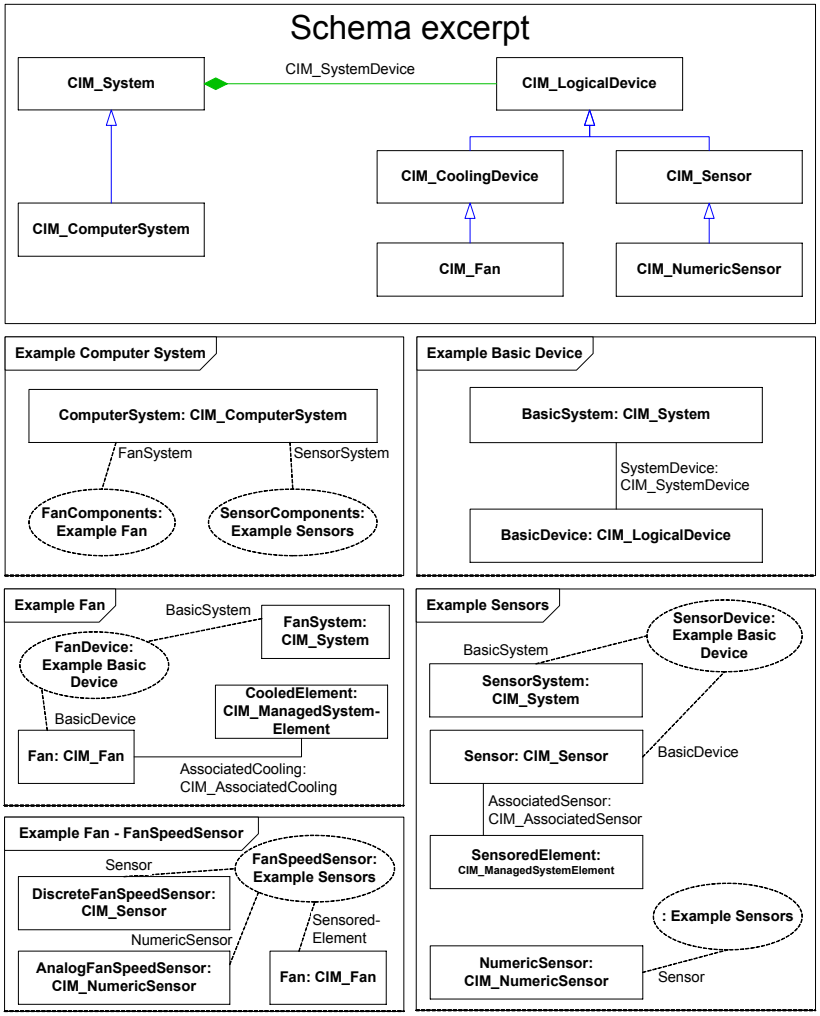


Figure 8 – Example for the determination of effective implementation requirements

The Example Computer System profile shown in Figure 8 is assumed to model systems that contain components such as fans or sensors. It references an Example Fan profile and an Example Sensors profile, and defines the ComputerSystem adaptation of the CIM_ComputerSystem class for the representation of computer systems. The ComputerSystem adaptation is based on both the FanSystem adaptation defined in the Example Fan profile and the SensorSystem adaptation defined in the Example Sensors profile. Not visible from the diagram – but assumed for this example – is that the reference to the Sensors profiles is optional, at this level leaving the decision about implementing this profile to the implementation.

Both the Example Fan profile and the Example Sensors profile in Figure 8 are assumed to be derived from an Example Basic Device profile. The Example Basic Device profile models basic systems, their devices and their relationships by means of the BasicSystem adaptation, the BasicDevice adaptation and the SystemDevice adaptation. It is important to realize that this relationship does not have to be repeatedly re-modeled in derived profiles.

The Sensor adaptation of the Example Sensors profile shown in Figure 8 is assumed to model sensors that report analogous values. The NumericSensor adaptation is based on the Sensor adaptation, and is assumed to model sensors that report discrete numeric values.

The Example Fan profile shown in Figure 8 is assumed to model a Fan adaptation for the representation of fans in systems. The Fan adaptation is based on the BasicDevice adaptation defined in the Example Basic Device profile. Furthermore, the Example Fan profile is assumed to define a conditional

FanSpeedSensor feature, with the condition stating that fans with sensors exist in the managed environment; consequently the FanSpeedSensor feature is only required to be implemented if that is the case.

The Example Fan profile is further assumed to refine the definitions of the Example Sensors profile with respect to fan speed sensors by defining the DiscreteFanSpeedSensor adaptation of the CIM_Sensor class based on the Sensor adaptation of the Example Sensors profile, and the AnalogFanSpeedSensor adaptation of the CIM_NumericSensor class based on the NumericSensor adaptation of the Example Sensors profile.

For this example the algorithm defined in 8.1.2 would be processed as follows:

- In step 1) the Example Computer System profile and the Example Fan profile are initially selected by the implementer, and added to the implementation profile set.
- In step 2)a), since all profiles reference the Profile Registration profile, that is added to the implementation profile set. Furthermore, since the Example Fan profile is based on the Example Basic Device profile, that profile is added to the implementation profile set.
- In step 2)b) in this example, it is assumed that no optional profiles are added.
- In step 2)c) the Example Sensors profile is added to the implementation profile set, since fans with sensors are assumed to exist in the managed environment.
- In step 3) another it is determined that another iteration over steps 2)a)-c) is required, since profiles (in this case only the Example Sensors profile) were added in the previous iteration.
- The second iteration of steps 2)a)-c) does not yield additional profiles.
- With step 3) determining that no futher iteration of 2)a)-c) is required, the implementation profile set is now complete, and contains all profiles that need to be implemented.

  In this example these are the profiles represented by a CSDs in Figure 8, and the Profile Registration profile. The implementation adaptation set does not yet contain any adaptations, because the resolution of conditions in step 2)c) did not require decisions about implementing adaptations.
- In step 4)a) mandatory adaptations are determined and added to the implementation adaptation set. For simplicity the considerations in this example are limited to the adaptations shown in Figure 8 :

  The Example Computer System profile requires the following mandatory adaptations:

  – ComputerSystem

  The Example Fan profile requires the following mandatory adaptations:

  – FanSystem and Fan.

  The Example Sensors profile requires the following mandatory adaptations:

  – SensorSystem, Sensor, AssociatedSensor and SensoredElement.

  The Example Basic Device profile requires the following mandatory adaptations:

  – System, BasicDevice and SystemDevice.
- In step 4)b) optional adaptations are determined and added to the implementation adaptation set. In this example it is assumed that the implementer decides not to add any optional adaptations.
- In step 4)c) conditional adaptations are determined and added to the implementation adaptation set. For simplicity of this example, only the AnalogFanSpeedSensor adaptation is further considered. The AnalogFanSpeedSensor adaptation had to be added because the managed environment in this example is assumed to contain fans with analog speed sensors.

- At this point the implementation adaptation set is complete, but still needs to be shrinked for the effects of basing adaptation on other adaptations.
- In step 5) the effective class adaptations are determined. Per definition, all leaf adaptations are effective class adaptations; in this example that are the following adaptations:
  - ComputerSystem adapting CIM_ComputerSystem, directly based on FanSystem and SensorSystem, and indirectly based on BasicSystem
  - Fan adapting CIM_Fan, and directly based on BasicDevice and on SensoredElement.
  - DiscreteFanSpeedSensor adapting CIM_Sensor, directly based on Sensor, and indirectly based on BasicDevice.
  - AnalogFanSpeedSensor adapting CIM_NumericSensor, directly based on NumericSensor, and indirectly based on Sensor and then on BasicDevice
  - AssociatedSensor adapting CIM_AssociatedSensor, with no base adaptations
  - SystemDevice adapting CIM_SystemDevice, with no base adaptations

  In this example there are no non-leaf adaptation with implementation requirements not addressed by a sub-adaptation.
- In step 6) the requirements of each effective class adaptations are enriched with those of their base adaptations. For example, the requirements of the ComputerSystem adaptation would be enrichted with those of the FanSystem adaptation, the SensorSystem adaptation and the BasicSystem adaptation.
- In step 7) all base adaptations that are not theirselves effective class adaptations are dropped from the implementation adaptation set. In this example these would be the BasicSystem adaptation, the BasicDevice adaptation, the SensorSystem adaptation, the Sensor adaptation, the SensoredElement adaptation, the NumericSensor adaptation and the FanSystem adaptation.
- In step 8) finally the requirements of the schema are merged into each effective class adaptation. For example, the ComputerSystem adaptation is enrichted with the requirements of the CIM_ComputerSystem class, the CIM_System class, and (not shown in Figure 8) the CIM_EnabledLogicalElement class, the CIM_LogicalElement class, the CIM_ManagedSystemElement class, and finally the CIM_ManagedElement class.

At this point, each adaptation in the implementation adaptation set defines its effective set of implementation requirements.

9 Profile specification requirements

This clause defines the requirements for profile specifications. Profile specifications are documents containing the definition of one or more profiles in textual form. This clause focuses on formal document aspects. Clause 7 describes profile definitions, focusing on profile content. All requirements stated in clause 7 for profile definitions apply correspondingly to profile specification documents.

9.1 General requirements

A profile specification published by DMTF shall conform to all requirements of this guide; in addition the requirements of ISO/IEC Directives, Part2 apply.

9.2 General conventions and guidelines

9.2.1 Notational conventions

This subclause defines notational conventions for profile specifications.

All words should be in lower case, and phrases should not be concatenated into one word unless one of the following conditions is met:

- The word starts a new sentence, heading or list item
- The word is a proper noun, such as for example Ethernet
- The word is an acronym, such as for example CPU
- The words is part of a profile name (see 7.4.2), such as for example Profile Registration
- The word is a feature name (see 7.12), such as for example FanStateManagement
- The word is an adaptation name (see 7.10.2.1), such as for example FanCapabilities
- The word is a schema element, such as for example CIM_SystemDevice, EnabledState or RequestStateChange( )

Elements of the managed environment and elements of the CIM model defined by the profile should be clearly distinguished in profile specifications. The following rule set is established in order to avoid wrong, unclear or confusing text that typically results from mixing elements from the managed environment and elements from the CIM model defined by a profile.

The following rules should be adhered with:

- CIM class names or adaptation names should not be used to refer to the object types defined in the management domain, and vice versa.
- CIM class names or adaptation names should not be used to refer to the managed objects in the managed environment (that are represented by their instances), and vice versa.
- References to instances of CIM classes or adaptations should contain the word "instance" unless the instance is clearly identified by an instance name.
- The managed object represented by an instance should be clearly identified, either immediately such as in "The VirtualSystem instance VSYS4 representing virtual system 4", or indirectly by a previously established context.
- The value of a property should be distinguished from the property itself.

For example, assume the specification of an Example Fan profile that defines a Fan adaptation of the CIM_Fan class. The Fan adaptation models fans that provide cooling for managed elements within systems. Further assume an example situation where a Fan instance named MAINCPUFAN represents the fan of the main CPU within an example system.

Table 2 – Specification recommendations

Recommended	Not recommended (wrong, unclear or confusing)
"The Fan instance MAINCPUFAN represents the CPU fan." NOTE This text defines MAINCPUFAN, such that it can be used in subsequent text. Typically definitions like this refer to a DMTF object diagram showing the identified instance.	"MAINCPUFAN is the fan of the main CPU." Problem: MAINCPUFAN identifies the Fan instance that *represents* the main CPU fan. Thus MAINCPUFAN is a CIM representation of the fan, but it *is not* the fan itself.
Preferred: "The value of the EnabledState property in MAINCPU-	"MAINCPUFAN is Enabled." Problem: CIM instances are not "Enabled"; instead CIM instances exhibit property values that reflect the state of the represented object in the managed environment. "The state of the main CPU fan is 2 (Enabled)."

FAN is 2 (Enabled)." Alternative: "The EnabledState value in MAINCPUFAN is 2 (Enabled)."	Problem: Confusion of the state of the managed object (the CPU fan) with the state as viewed through the CIM instance representing the managed object. If the CPU fan is enabled, that is reflected in the Fan instance MAINCPUFAN through the value 2 (Enabled) for the EnabledState property.
	"The fan state is Enabled." Problem: Confusion of the state of the managed object and the state of the instance representing the managed object.
	"EnabledState is 5." Problem: Property and property value are not distinguished.

9.2.2 Conventions and guidelines for diagrams in profile specifications

Four types of diagrams are commonly used in profile specifications:

- **DMTF collaboration structure diagrams** show the structure of a profile or subset thereof, and the collaborations that this structure makes possible.
- **DMTF class diagrams** show the classes adapted by a profile (and possibly classes adapted by related profiles)
- **DMTF object diagrams** (also referred to as instance diagrams) show a set of related objects (or instances of classes) at a point in time. Object diagrams may be associated with use cases – showing how the use case affects properties and classes.
- **DMTF sequence diagrams** show the interaction between adaptation instances in terms of methods and operations.

Table 3 lists the requirements for these types of diagrams if used within profile specifications. Other types of diagrams may be used in profile specifications. Note that the usage requirements depend on the structure chosen for the profile specification; see 9.3 for a definition of profile specification structures.

Table 3 – Profile diagram types

Diagram type	Usage requirements		Description
	Tradional structure	Condensed structure	
DMTF collaboration structure	Optional	Required	See 9.2.2.3 .
DMTF class	Required	Optional	See 9.2.2.4 .
DMTF object	Optional	Optional	See 9.2.2.5 .
DMTF sequence	Optional	Optional	See 9.2.2.6 .

9.2.2.1 General diagram guidelines

Diagrams are not normative; all normative information shall be provided in text.

Fonts in diagrams should not be less than 10 point, and shall not be less than 6 point.

9.2.2.2 Notational and color conventions

For DMTF diagrams the notational conventions as established by the OMG UML Superstructure apply.

In addition, the color conventions as defined in this subclause should be applied for DMTF class diagrams, DMTF object diagrams and DMTF sequence diagrams. Deviations from the color conventions are

permitted, but shall be documented in the profile specification and consistently applied within one profile specification.

The conventions defined in this subclause are an adapted subset of the conventions outlined in diagrams that depict schema definitions owned by DMTF.

The following color conventions apply:

- Associations – red line

- Aggregation association – green line with a hollow diamond at the aggregating end

- Composition association – green line with a solid diamond at the aggregating end

- Inheritance relationships – blue line with hollow arrow at the superclass end

Deprecated content - start

- Composition association – green line with a hollow diamond and a dot at the aggregating end

NOTE In OMG UML Superstructure a dot at the endpoint indicates that the endpoint is owned by the connected element. Since with CIM associations an association endpoint is owned by the association itself, the former convention of showing a dot is incorrect.

- Inheritance relationships – blue line with solid arrow at the superclass end

NOTE In OMG UML Superstructure a closed arrow at an endpoint of a UML graphic path is defined to indicate an UML extension, whereas a hollow arrow is defined to indicate a UML generalization. Since CIM inheritance is logically equivalent to the UML concept of generalizations - and not to that of UML extensions – a hollow arrow is required at the end connecting to the generalized element, whereas the former use of a solid arrow is incorrect.
A UML extension indicates that the properties of a metaclass are extended through a stereotype to flexibly add (and later remove) stereotypes to classes. A UML generalization is a taxonomic relationship between a more general classifier and a more specific classifier where each instance of the specific classifier is also an indirect instance of the general classifier, and the specific classifier inherits the features of the more general classifier.

Deprecated content - end

- Deprecated class, property or method – the letter D in curly brackets:

  {D}

- Experimental class, property or method - the letter E in curly brackets:

  {E}

The deprecation and experimental indicators reflect the specification of the corresponding qualifiers in the schema definition.

NOTE The deprecation and experimental indicators represent schema definitions only. They are not used in the case where subsequent versions of profiles mark profile defined elements (such as class adaptations or references to properties or methods) that were introduced in previous versions of the profile, as deprecated or experimental.

9.2.2.3 DMTF collaboration structure diagram guidelines

DMTF collaboration structure diagrams show the structure of a complete profile, or a logically related subset of profile elements (such as for example a feature), and all or a part of the collaboration defined by the profile.

DMTF collaboration structure diagrams are a specialization of UML composite structure diagrams; for the normative definition of UML composite structure diagrams see OMG UML Superstructure.

For DMTF collaboration structure diagrams (CSDs) the following additional rules and conventions apply:

- A CSD shall depict either the complete collaboration defined by a profile, or a subset of that collaboration.
- A CSD shall be labeled as follows:

```
CSDLabel = ProfileName [ "-" SubpartName [ SubpartType] ]
```

  `SubpartName` shall only be used if the CSD shows a subcollaboration of the profile; in this case the `SubpartType` may identify the type of the subpart, such as for example a feature, pattern or scenario.

- Adaptations of ordinary classes or indications shall be represented as UML parts. Each UML part shall be shown as solid rectangles (boxes), and shall be named as follows:

```
PartName = AdaptationName ":" ClassName [ "[" PartMultiplicity "]" ]
```

  UML part multiplicities shall correspond to the number of instances required by an adaptation. UML part multiplicities shall be shown if deviating from the default "*" (zero to many).

- Adaptations of associations shall be represented by UML connectors. Each UML connector shall be shown as a solid line, connecting two UML parts. Each UML connector shall be named as follows:

```
ConnectorName = AdaptationName ":" AssociationClassName
```

  – References defined by association adaptations may be represented as UML endpoint names. If used, UML endpoint names shall be shown as text at the ends of a UML connector.

  – Reference multiplicities may be represented by UML endpoint multiplicities. Reference multiplicities shall be represented as UML endpoint multiplicities if deviating from the default multiplicity "*" (zero to many).

- The use of a profile may be represented as UML collaboration use. UML collaboration uses shall be shown as dashed ovals. Each UML collaboration use shall be named as follows:

```
CollaborationUseName = [ ProfileReferenceName ] ":" ProfileName
```

  `ProfileReferenceName` shall be the name of the profile reference as defined by the referencing subject profile

  `ProfileName` shall be the name of the related profile or the name of the subject profile in the case where the subject profile defines adaptations based on other adaptations in the same profile. If in the latter case a `ProfileReferenceName` is specified, the UML collaboration use

represents a complete new use of the subject profile by itself; otherwise, the UML collaboration use serves only as an anchor point for base adaptations.

- The relationship between an adaptation of an ordinary class defined in the subject profile and profiles defining base adaptations of that adaptation shall be shown as UML role bindings. Each UML role binding shall be shown as dashed line connecting a UML collaboration use representing (the collaboration defined by) a profile that defines a base adaptation, and the UML part representing a class adaptation defined in the subject profile. Each UML role binding shall be labeled close to the class adaptation end, as follows:

  ```
  EndRoleName = BaseAdaptationName
  ```

  `BaseAdaptationName` shall be the name of the base adaptation.

Figure 9 shows examples of three DMTF collaboration structure diagrams depicting collaborations defined by one autonomous profile and two component profiles.

NOTE The dash-dotted separation lines are not part of the conventions for DMTF collaboration structure diagrams as defined in this subclause; they are shown here such that multiple DMTF collaboraration structure diagrams can be condensed into one diagram.

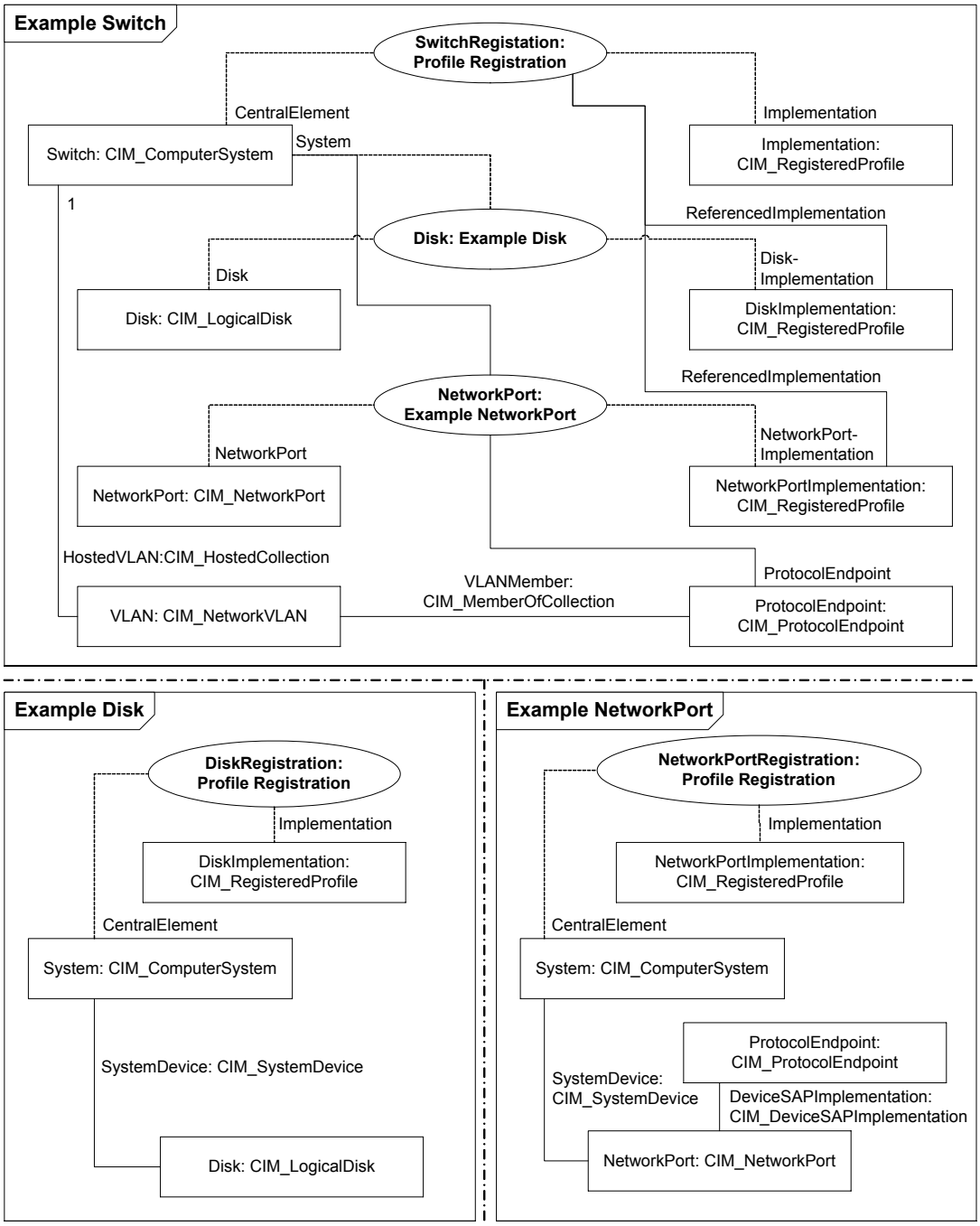


Figure 9 – Example of a DMTF collaboration structure diagram

The upper part of Figure 9 shows the collaboration defined by an autonomous Example Switch profile. The Example Switch profile models a switch with switch ports and with a disk that contains configuration data. The collaboration defined by the autonomous Example Switch profile is depicted as follows:

- The Example Switch profile defines a Switch adaptation of the CIM_ComputerSystem class. This is depicted by the UML part (solid rectangle) named "`Switch:CIM_ComputerSystem`".
- The Profile Registration profile is referenced by the Example Switch profile. This is depicted by the UML collaboration use (dashed oval) named "`SwitchRegistration: Profile Registration`".

- The System adaptation is based on the CentralElement adaptation of the Profile Registration profile. This is depicted by the UML role binding (dashed line) named `CentralElement` that connects the UML part named "`Switch:CIM_ComputerSystem`" with the UML collaboration named "`SwitchRegistration: Profile Registration`".
- The Example Switch profile references the Example Disk profile and the Example Network Port profile. This is shown by the UML collaboration uses (dashed ovals) named "`Disk: Example Disk`" and "`NetworkPort: Example NetworkPort`".
- The Profile Registration profile requires profiles to express profile dependencies by means of the CIM_ReferencedProfile association. For example, for the Example Disk profile this is depicted by the UML role binding named `ReferencedImplementation` connecting the UML collaboration named "`SwitchRegistration: Profile Registration`" with the UML part (solid rectangle) named "`DiskImplementation: CIM_RegisteredProfile`". The latter corresponds to the DiskImplementation adaptation of the Example Disk profile, as depicted by the UML role binding named `Implementation` connecting it with the UML collaboration use named "`Disk: Example Disk`".
- The Example Switch profile defines a VLAN adaptation of the CIM_NetworkVLAN class. This is depicted by the UML part named "`VLAN: CIM_NetworkVLAN`".
- The Example Switch profile defines a HostedVLAN adaptation of the CIM_HostedCollection association for the representation of the relationship between a switch and the VLANs hosted by that switch. This is depicted by the UML connector (solid line) named "`HostedVLAN: CIM_HostedCollection`".
- Note that the UML endpoint multiplicity at the Switch side is 1, indicating that the VLAN adaptation relates to the VLAN endpoints of exactly one switch. If the VLAN ranges over several switches, the VLAN elements hosted by the other switches would have to be provided by separate VLAN instances. This behavior is also implied by the definition of the CIM_NetworkVLAN class.
- Note that the implied UML part multiplicity of the "`Switch: CIM_ComputerSystem`" UML part is "*", indicating that an implementation of the Example Switch profile controls zero or more switches.

9.2.2.4 DMTF class diagram guidelines

9.2.2.4.1 General requirements

DMTF class diagrams are UML class diagrams (see OMG UML Superstructure) that conform to additional requirements defined in this subclause.

DMTF class diagrams shall show classes and associations.

NOTE A particular class may be shown multiple times in a class diagram; this is in conformance with the rules for UML diagrams specified in OMG UML Superstructure.

DMTF class diagrams shall not mix the conventions of class and object diagrams.

DMTF class diagrams may show properties and methods; if so, only properties and methods referenced by the subject profile should be shown.

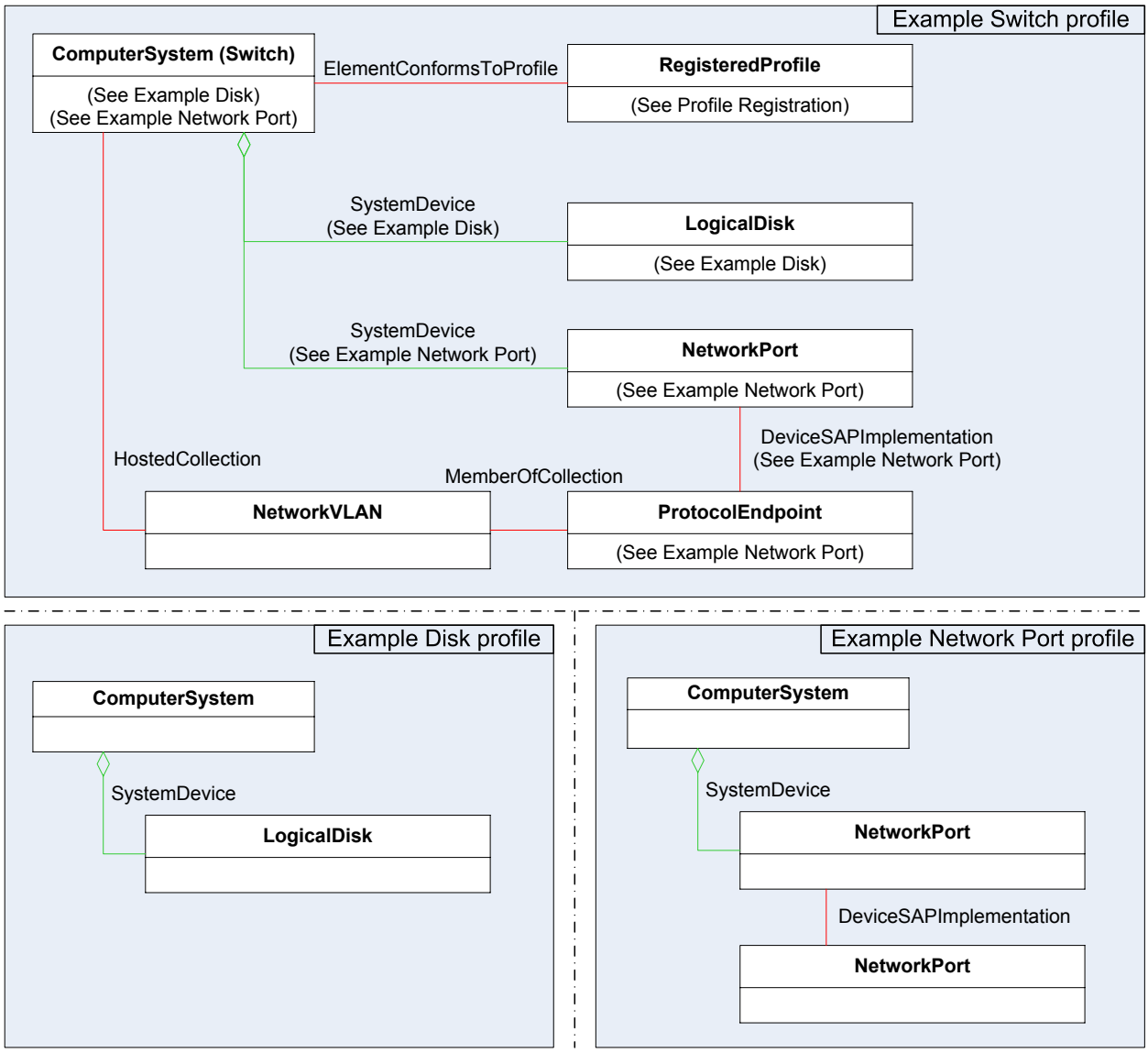


Figure 10 – Examples of DMTF class diagrams

Figure 10 shows examples of profile class diagrams from one autonomous profile profile and two component profiles.

NOTE The shaded rectangles and the dash-dotted separation lines are not part of the conventions for DMTF class diagrams as defined in 9.2.2.4; they are shown here such that multiple DMTF class diagrams can be condensed into one diagram.

The upper part of Figure 10 shows the profile class diagram of an autonomous Example Switch profile. It is assumed that the central class adaptation of the Example Switch profile is the Switch adaptation that is based on the CIM_ComputerSystem class, and in addition is based on both the ComputerSystem adaptations defined in the Example Disk profile and in the Example Network Port profile.

Deprecated content - start

9.2.2.4.2 Specific requirements for DMTF class diagrams in traditional profile specifications

Minor revisions of profiles initially specified in compliance with version 1.0 of this guide may continue using profile class diagrams as defined in this subclause.

The requirements in this subclause apply in addition to those specified in 9.2.2.4.1.

Each profile specification in profile specifications applying the traditional profile structure shall contain one DMTF class diagram that depicts the central elements of the management interface defined by the subject profile by showing classes and associations that are adapted by the subject profile or by a related profile (see 7.6). That DMTF class diagram shall have a label formatted as follows:

```
DiagramLabel = ProfileName ": Profile class diagram"
```

The schema prefix (like for example "CIM_") shall be omitted from names of classes defined in a DMTF maintained CIM schema. Prefixes should be shown if the profile adapts classes that are not defined in a by DMTF maintained CIM schema.

Classes adapted by the subject profile shall be represented with a box that exhibits two horizontal compartments.

The top compartment shall contain the adaptation name as defined in 7.10.2.1, including the case where the name is in the deprecated format using a class name and an optional modifier.

If a subject profile refers to a class adaptation defined in a referenced profile, the lower compartment shall contain the string:

```
Reference = "(See " RegisteredProfileName [ "profile" ] ")"

ProfileDesignator = ScopingProfileDesignator /

ReferencingProfileDesignator / SpecificProfileDesignator

ScopingProfileDesignator = "scoping profile"

ReferencingProfileDesignator = "referencing profile"

SpecificProfileDesignator = RegisteredProfileName [ "profile" ]
```

`RegisteredProfileName` is the registered profile name of the referenced profile.

The depiction of adaptations shall not include properties or methods. Inheritance should only be shown if the profile adapts a class and its superclass.

NOTE Eliminating properties and methods eliminates the risk that these elements are specified differently in the diagram and the text format included in profile specifications.

The depiction of an association shall be labeled with the association adaptation name. If the adaptation of an association is defined by a referenced profile, the label for that association shall contain a reference to the referenced profile, using the format defined by the `Reference` ABNF rule.

If a profile defines multiple adaptions of the same adapted class for multiple purposes, then each adaptation should be shown separately.

The depiction of association adaptations shall show multiplicities. Note that these multiplicities are the multiplicities as exposed by the association adaptation, can be constrained beyond those defined for the adapted association in the schema. For example, if a profile in an association adaptation requires a multiplicity of 1-n, but the schema defined multiplicity is 0-n, then the multiplicity shown in the class diagram shall reflect the narrowed multiplicity required by the association adaptation.

Deprecated content - end

9.2.2.5 DMTF object diagram guidelines

DMTF object diagrams (also referred to as instance diagrams) are UML object diagrams (see OMG UML Superstructure) that satisfy the additional constraints defined in this subclause.

DMTF object diagrams shall show a set of related adaptation instances at a point in time. DMTF object diagrams may be associated with use-cases – showing how adaptation instances, particularly their property values and their relationships, are visible to clients in the process of performing a sequence of activities as described by a use-case.

DMTF object diagrams depict example instantiations and should illustrate best practice implementations.

The labels of any CIM instances in a DMTF object diagram shall be specified using the format (in ABNF):

```
InstanceLabel = [ InstanceName ] " : " ClassAdaptationName
```

The `ClassAdaptationName` ABNF rule shall evaluate to the name of a class adaptation defined in the subject profile or a referenced profile. The value of the `InstanceName` ABNF rule is an arbitrary string that may be used to refer to the instance from any text describing the diagram; it may be omitted if the resulting label is not ambiguous within the diagram.

Instances of abstract classes shall not be shown in DMTF object diagrams. If a variety of concrete subclasses are applicable in a particular case, a concrete subclass shall be selected and explanatory text be provided with the diagram stating that the other concrete classes are applicable as well.

Instances shall be represented with a box that exhibits the two horizontal compartments. The top compartment shall contain the instance label as defined for the `InstanceLabel` ABNF rule. The bottom compartment may contain applicable properties that are needed to be illustrative, including properties that are defined in the schema definition of adapted classes, but are not referenced by the subject profile or a referenced profile.

For each applicable property the property name and its value shall be listed using the format (in ABNF):

```
PropertyEntry = PropertyName PropertyAssignment PropertyValue

PropertyName = IDENTIFIER

PropertyValue = initializer

PropertyAssignment = "="
```

Deprecated content – start

Minor revisions of profiles initially specified in compliance with version 1.0 of this guide may continue using the colon as the assignment operator in property entries.

```
PropertyAssignment = "=" | ":"
```

Deprecated content – end

Methods should not be shown in DMTF object diagrams.

If UFiT values are included in the object diagram, they should conform to DSP0215.

DMTF object diagrams shall be accompanied by descriptive text that explains the diagram and its pertinence.

Associations shall be depicted as UML links. Associations with properties other than reference properties may be depicted as a separate UML object that contains the properties and is connected to the association link with a dashed line.

Deprecated content - start

Minor revisions of profiles specified in compliance with version 1.0 of this guide may continue depicting association properties as a list below the association class name.

Deprecated content – end

DMTF object diagrams shall be accompanied by descriptive text that explains the diagram and its pertinence.

9.2.2.6 DMTF sequence diagram guidelines

DMTF sequence diagrams are UML sequence diagrams (see OMG UML Superstructure) that satisfy the additional constraints defined in this subclause.

DMTF sequence diagrams shall depict the interaction between CIM instances, in the form of method or operation calls and call returns.

Object names should be specified using the format:

```
[ InstanceName ] ":" AdaptationName
```

The `AdaptationName` shall name a class adaptation defined by the profile unless a class adaptation was not defined by the profile. The `InstanceName` may be omitted if the resulting object name is not ambiguous.

9.2.2.7 Deprecated profile elements and deprecated schema elements in diagrams

Revisions of profiles may deprecate profile elements defined in a previous version. Diagrams in profile specifications may include deprecated profile elements; if so, they shall be shown marked as deprecated as required by the notational and color conventions; see 9.2.2.2 .

Profiles may refer to deprecated schema elements such as for example classes (see 7.10.2.2), references to properties (see 7.10.2.5) or references to methods (see 7.10.3.1). Diagrams in profile specifications may include deprecated schema elements; if so, they may be shown marked as deprecated as required by the notational and color conventions; see 9.2.2.2 .

9.2.3 Conventions for the specification of requirement levels

In profile specifications requirement levels (see 7.1) are stated using keywords as defined in this subclause.

- The mandatory requirement level (see 7.1.1) shall be stated using the keyword "mandatory".
- The conditional requirement level (see 7.1.2) shall be stated using the keyword "conditional".
- The optional requirement level (see 7.1.3) shall be stated using the keyword "optional".

9.2.4 Conventions for the specification of conditional elements

This subclause defines requirements for the specification of conditional elements in profile specifications.

9.2.4.1 General

Conditions shall be defined using one of the mechanisms defined in 7.2.

9.2.4.2 Conventions for the specification of conditional elements outside of tables

In any text outside of tables the fact that an element is defined as conditional shall be phrased as follows,

```
ConditionalPhrase = "The implementation of the" ElementName ElementType
"is conditional."

ElementName = PROFILE_IDENTIFIER | IDENTIFER
```

; shall identify the conditional element

```
ElementType = "profile" | "feature" | "adaptation" | "property" | "method"
| "parameter"
```

In cases where it is not possible to apply this phraseology, alternatively a condition and its consequence may be stated as a conditional sentence in English language.

The text defining the condition shall be phrased in the format of a `ConditionStatement` as detailed below:

```
ConditionStatement = "Condition:" ConditionSpecification
```

`ConditionSpecification` shall be an appropriate textual representation of the basic types of conditions and their combination using Boolean operators, as specified in 7.2 .

Examples:

- "Condition: The Fan adaptation is implemented."
- "Condition: The FanSpeedSensor feature is implemented."
- "Condition: The managed environment contains fans with simple sensors or the managed environment contains fans with numeric sensors."
- "Condition: Any of the following:
  - The managed environment contains fans with simple sensors
  - The managed environment contains fans with numeric sensors."

9.2.4.3 Conventions for the specification of conditional elements within tables

Within tables, a conditional element shall be designated with the word "Conditional" (without additional text) within the table column indicating the requirement level, as follows:

```
ConditionInTable = "Conditional" [ NL "(" ConditionSpecification [ Refer-
ence ] ")" ]
```

`NL` introduces a new line. `ConditionSpecification` shall be a statement that is either true or false. `Reference` may refer to a separate clause detailing the condition.

Alternatively, the condition may be specified in a corresponding cell within a description column, cell as defined in 9.2.4.2. If the text in the Description would exceed 20 words, it shall be replaced by a reference to a separate subclause that defines the condition as required by 9.2.4.2 .

An example of the specification of a condition within a table is given in Table X-1.

9.2.5 Conventions for the specification of value constraints

A profile may constrain property values or method parameter values to a single value or a set of values.

For string typed properties and parameters profiles are required to specify a mechanism that conveys the format used for their values (see 7.10.2.6).

Table 4 provides examples of applications of the provisions in this subclause.

If in a profile specification a format specification is stated in the form of a regular expression, that shall be preceded by an equivalent format definition stated in the form of normative text. The regular expression based format definition shall follow encompassed by brackets. Within the brackets the keyword "pattern" shall be used to identify the regular expression, followed by the regular expression as a quoted string and compliant with the regular expression syntax defined in ANNEX B. For an example, see PermanentAddress in Table 4 .

NOTE Regular expressions can be used in code that validates formats. Textual descriptions provide equivalent information suitable for human readers.

Within tables, the name of the property or parameter is listed under a separate column, and the value constraint shall be expressed within the corresponding cell of the Description column in the form of the a normative statement, as follows:

- If the value set for a string property or parameter is constrained to just one value, that value shall be stated and a regular expression pattern should not be specified. For an example see OtherPortType in Table 4 .
- For the specification of value set of properties or parameters without a Values qualifier, a requirement for exactly one valid value shall be specified as follows: "Value shall be" or "Value shall match", followed by the value. For an example see PortNumber in Table 4 .
- For the specification of the value set of properties or parameters without a Values qualifier, a requirement for a list of valid values shall be specified as follows: "Value shall match", followed by a list of values separated by vertical bars. For an example see SupportedMaximumTransmissionUnit in Table 4 .
- For the specification of the value set of properties or parameters with a Values qualifier, a single valid value shall be specified as "Value shall be" or "Value shall match", followed by the element from ValueMap value set and followed by the parenthesized corresponding (textual) element of the Values value set. For an example see PortType in Table 4 .
- For the specification of the value set of a properties or parameters with a Values qualifier, a list of valid values shall be specified as "Value shall match", followed by a list of elements from the ValueMap value set separated by vertical bars and followed by a parenthesized list of corresponding elements from the Values value set separated by "or". For an example see LinkTechnology in Table 4 .

NOTE The lists of values from the ValueMap value set and from the Values value set are specified separately. This allows the ValueMap value list to be a valid regular expression, enabling automatic generation of profile specification tables from a separate source (such as XML) that can also be used for testing. If elements from the ValueMap value set and the Values value set were mixed (like for example, "ProtocolIFType matches 4096 (IP v4) | 4097 (IP v6), | 4098 (both)"), then the result is not a valid regular expression.

Outside of tables, value constraints shall be expressed in form of normative sentences, such as for example: "The value of the BlockSize property shall convey the formatted block or sector size, and shall always be 512.". The examples listed above for the definition of value constraints within tables apply correspondingly, for example replacing the phrase "Value shall …" with the phrase "The value of the xxx property shall …".

Some CIM classes define a separate property for the specification of valid formats of the value of another property. The second adaptation example in Table 4 shows a format definition for the Name property in a StorageVolume adaptation of the CIM_StorageVolume class with valid formats conveyed through the value of the NameFormat property.

Table 4 – Example of string property format definition

X-7 Implementation

…

X-7.4 Adaptation: VirtualNetworkPort: CIM_NetworkPort

This subclause defines the adaptation of the CIM_NetworkPort class for the representation of network ports in virtual systems.

X-7.4.1 Implementation requirements

Table X-11 lists the implementation requirements for the VirtualNetworkPort adaptation.

Table X-11 – Adaptation: VirtualNetworkPort: CIM_NetworkPort

Element	Requirement	Description
…	…	…
UsageRestriction	Mandatory	Value shall be 2 (Front-end-only)
PortType	Mandatory	Value shall match be 1 (Other)
OtherPortType	Mandatory	Value shall be "Dynamic port"
PortNumber	Mandatory	Value shall be 0
LinkTechnology	Mandatory	Value shall match 2 | 3 | 5 ("Ethernet" or "IB" or "FDDI")
PermanentAddress	Mandatory	Value shall be formatted as 16 consecutive uppercase hexadecimal digits (pattern "^[0123456789ABCDEF]{16}$")
SupportedMaximumTransmissionUnit	Mandatory	Value shall be 1526 | 4096
…	…	...

…

X-7.6 Adaptation: StorageVolume: CIM_StorageVolume

X-7.6.1 Implementation requirements

Table X-12 lists the implementation requirements for the StorageVolume adaptation.

Table X-12 – Adaptation: StorageVolume: CIM_StorageVolume

Element	Requirement	Description
…	…	…
Name	Mandatory	See X-7.6.2 .
NameFormat	Mandatory	Value shall be 7 (SNVM), 8 (NodeWWN) or 9 (NAA)
…	…	...

…

X-7.6.2 Property: Name

Valid formats of the Name property are constrained by the value of the NameFormat property, as follows:

- If the value of the NameFormat property is 7 (SNVM), the value of the Name property shall convey the vendor name, product name and serial number of the storage volume as three strings separated by "+" characters. The vendor name shall have exactly 8 characters and the product name shall have exactly 16 characters. Both names may contain blanks as significant characters and if necessary shall be padded with blanks to match the required length. The serial number shall be formatted using uppercase hexadecimal digits (pattern "^[A-Za-z]{8}\+[A-Za-z]{16}\+ [0123456789ABCDEF]*$").
- If the value of the NameFormat property is 9 (NAA), the value of the Name property shall convey the systems hardware ID as specified in T10 SPC and shall be formatted as 16 consecutive uppercase hex digits (pattern "^[0123456789ABCDEF]{16}$")
- If the value of the NameFormat property is 8 (NodeWWN), the value of the Name property shall convey the systems Fibre Channel WWN and shall be formatted as 8 consecutive uppercase hex digits (pattern "^[0123456789ABCDEF]{8}$")

…

9.2.5.1 Conventions for the specifications of default property values

If a profile defines a default value for a property (see 7.10.2.5), that shall be specified using the following format:

```
PropertyDefaultValuePhrase = "Default value is" value "."
```

9.2.5.2 Conventions for the specification of reference multiplicities

The specification of references in association adaptations shall include text specifying the multiplicity of the reference if the schema defined multiplicity is further constrained by the profile; see 7.10.2.5 .

The format is

```
MultiplicitySpecification = "Multiplicity:" MultiplicityValue
```

Deprecated content – start

Minor revisions of profiles initially specified in compliance with version 1.0 of this guide may continue using the word "cardinality" in place of "multiplicity".

Deprecated content - end

`MultiplicityValue` shall specify the multiplicity, as follows:

1 indicates that exactly one instance is referenced

* indicating 0 or more instances are referenced

m..n indicating *m* to *n* instances are referenced, where *m* is 0 or a positive integer and *n* is a positive integer or * representing unlimited.

If no multiplicity is specified in the profile, the multiplicity defined in the schema definition of the reference applies; this may be emphasized by explicitly stating "Reference multiplicity conforms to the schema definition".

Note that multiplicities of references are specified in the context of a class adaptation, and that multiplicities of references in different adaptations of the same association may be different.

9.3 Profile specification structures

9.3.1 General

This guide defines a choice of two structures for profile specifications: The condensed structure and the traditional structure.

The condensed profile specification structure should be favored for new profile specifications that are originally created in conformance to this guide.

Revisions of existing profiles may continue to use the traditional structure, and may apply a mixture of both structures with respect to the definition of indications.

NOTE The last rule was established to enable revisions of existing profiles to conform with provisions defined by this guide with respect to the definition of indication requirements, without requiring these revisions having to conform with other provisions of this guide.

9.3.2 Condensed profile specification structure

The condensed profile specification structure provides for a comprehensive definition of class adaptations as part of the Implementation clause; thus it condenses information into the Implementation clause that

with version 1.0 of this guide was spread over the "CIM elements" clause, the Methods clause and the Implementation clause.

In the condensed profile specification structure, the location for the table listing all class adaptations defined by a profile is in the Synopsis clause. This enables a straight forward definition of class adaptations with a direct entry path through the Synopsis clause that provides the overview information, and tables with forward references to subclauses of the Implementation clause that provide detailed implementation information.

Deprecated content - start

9.3.3 Traditional profile specification structure

Minor revisions of profiles initially specified in compliance with version 1.0 of this guide may continue using the traditional profile specification structure as defined in this subclause.

The traditional profile specification structure originally defined in version 1.0 of this guide spreads the entry information to a profile over the Synopsis clause and the CIM Elements clause. The CIM Elements clause typically contains back references to subclauses of the Implementation and the Methods clause that provide detail information.

In this version of this guide the traditional structure was established to allow for revisions of existing profile specifications originally created in conformance with version 1.0 of this guide to remain compliant to this guide without structural changes.

Revisions of existing profiles may continue to use the traditional structure, and may apply a mixture of both structures with respect to the definition of indications.

Deprecated content - end

9.3.4 Usage of profile specification structures

The two profile specification structures are depicted in Figure 11.

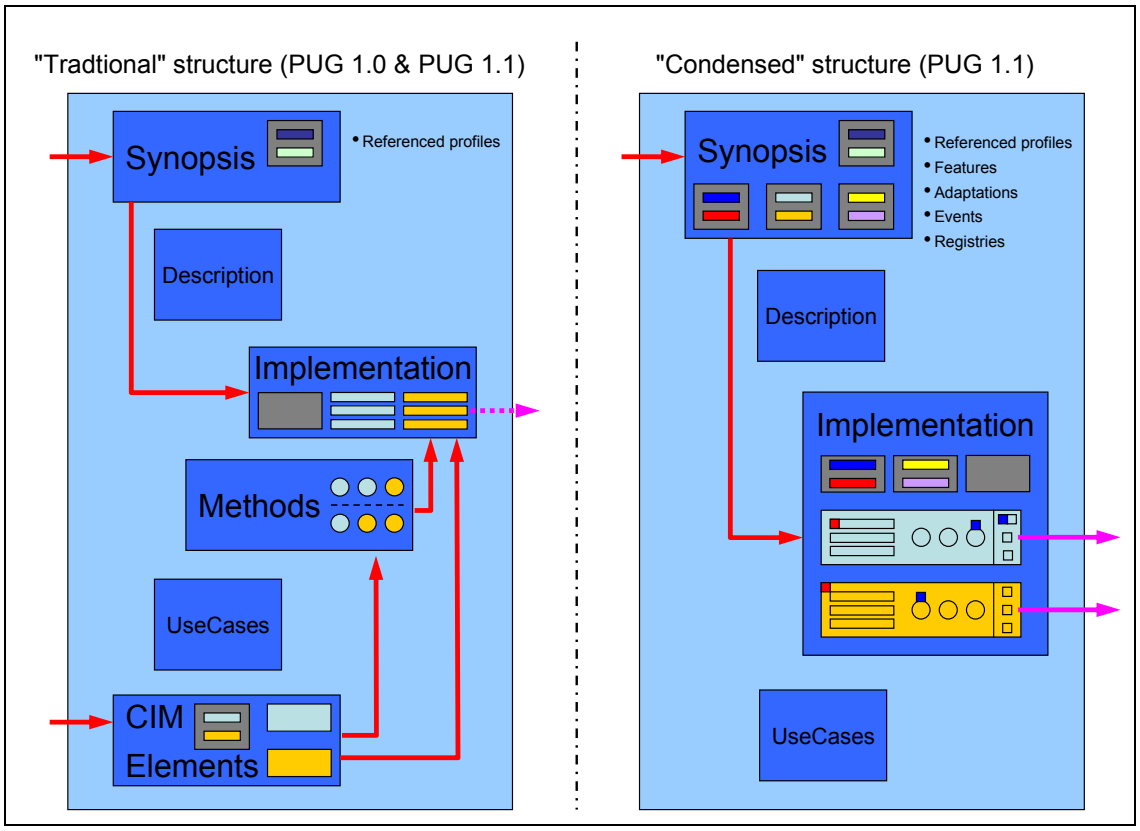


Figure 11 – Traditional and condensed profile structures

On the left side of Figure 11 the major clauses are shown with the traditional profile structure applied. Note the two entry paths into the profile, one following through the Synopsis clause, the other one following through the "CIM elements" clause. On the right side of Figure 11 the major clauses are shown with the condensed profile structure applied. Note that there is only one entry path into the profile, and that adaptations are comprehensively organized within the Implementation clause, with all pertinent information required for the implementation of a particular adaptation presented within one subclause. The blue and red colored squares indicate that the implementation of some elements is only required as the "blue" or the "red" feature are implemented.

9.4 Requirements for profile specification clauses

9.4.1 General

The requirements for profile specification clauses differ with the structure chosen for the subject profile; see 9.3 . Table 5 lists the profile specification clauses in the order they shall appear in profile specifications, along with references to subclauses of this guide or documents referenced by this guide that detail the requirements for the specification of respective clauses in profile specifications.

Table 5 – Requirements for profile specification clauses

Clause name	Condensed structure	Traditional structure
Scope	Required, see ISO/IEC Directives, Part2, 6.2.1 .	
Normative references	Required, see ISO/IEC Directives, Part2, 6.2.2 .	
Terms and definitions	Required, see 9.4.3 .	
Symbols and abbreviated terms	Required, see ISO/IEC Directives, Part2, 6.3.2 .	

Conformance	Optional, see 9.4.4 .	
Synopsis	Required, see 9.4.3 . Requirements differ based on the chosen structure.	
Description	Required, see 9.4.6 .	
Implementation	Required, see 9.4.7 . Requirements differ based on the chosen structure.	
Methods	Not allowed, content covered in Implementation clause; see 9.4.7 .	Required, see 9.4.8 .
Use cases	Required, see 9.4.9 .	
CIM elements	Not allowed, content covered in Implementation clause; see 9.4.7 .	Required, see 9.4.10 .

Spelling of clause names and subclause names shall follow normal English grammar rules. Arbitrary capitalization of words should be avoided.

9.4.2 Requirements for the numbering of profile specification clauses and subclauses

ISO/IEC Directives, Part2 requires clauses and subclauses to be numbered.

An organization may opt to "demote" the clauses to subclauses at a lower heading level. For example, Clause "6 Synopsis" may become subclause "8.6 Synopsis" or "8.2.6 Synopsis" within a larger aggregating document. However, the relative heading numbering shall be maintained at respective lower levels (that is, all headings are demoted by the same number of heading levels), and all clauses starting with the Synopsis clause shall be provided. This allows embedding profile specifications in a larger document while preserving a recognizable profile format for readers.

9.4.3 Requirements for the specification of the "Terms and definitions" clause

Each profile specification shall have a "Terms and definitions" clause.

The "Terms and definitions" clause shall be specified as defined in ISO/IEC Directives, Part2, 6.3.1 and Appendix D .

NOTE ISO/IEC Directives, Part2 and other ISO documents establish rigid rules with respect to the capitalization of terms. Generally, terms are required to be in lowercase unless otherwise required by English grammar rules.

The "Terms and definitions" clause shall contain the text stated in Table 6 immediately after the heading.

Table 6 - Common text for the "Terms and definitions" clause of profile specifications

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. 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 guide, clauses, subclauses or annexes indicated with "(informative)" as well as notes and examples do not contain normative content. The terms defined in DSP0004 and DSP0223 apply to this profile.

Profiles referencing operations defined in DSP0200 instead of DSP0223 shall refer to the terms defined in DSP0200 instead of those defined in DSP0223.

9.4.4 Requirements for the specification of the Conformance clause

The speficication of a conformance clause is optional.

Generally, the conformance definitions defined by this guide for implementations (see) apply to profiles. Profiles may specify additional conformance rules.

9.4.5 Requirements for the specification of the Synopsis clause

This subclause defines requirements for the Synopsis clause in profile specifications.

9.4.5.1 General

Each profile specification shall have a Synopsis clause.

The Synopsis clause of a profile specification shall conform to the rules defined in subclauses 9.4.5.2 to 9.4.5.5 .

9.4.5.2 Requirements for the specification of profile attributes

Profile attributes shall be listed first in the Synopsis clause. If other subclauses are defined within the Synopsis clause, the profile attributes shall be listed in a separate subclause of the Synopsis clause named "Profile attributes"; otherwise, the profile attributes shall be listed directly within the Synopsis clause immediately following the heading of the Synopsis clause.

NOTE ISO/IEC Directives, Part2 require that no normative text is put at the beginning of a clause if that clause contains subclauses (avoidance of "hanging" definitions); this is the reason for requiring separate subclauses in the case any subclause is defined within the Synopsis clause. Such subclauses might be required for example because table cell space requirements are exceeded in tables required by other subclauses of 9.4.5, or because the definition of the scoping algorithm requires a separate subclause .

Profile attributes shall be listed as a sequence of attribute statements. These statements should be placed first in the Synopsis clause; they should not be placed in a table.

The sequence of attribute statements and their format is defined by the "Attribute statement" column of Table 7; corresponding values in the Requirements column refer to subclauses of clause 7 that provide details about the respective profile attributes.

Table 7 – Requirements for the specification of profile attributes

Attribute statement	Requirements
Profile name: <registered profile name>	Required; see 7.4.2 .
Version: <registered profile version>	Required; see 7.4.3 .
Organization: <registered organization name>	Required; see 7.4.4 .
Abstract indicator: <abstract profile indicator>	Optional; see 7.6.4 . Value shall be "true" for abstract profiles, and "false" otherwise. Default: "false".
Profile type: "autonomous" or "component"	Required; see 7.6.5.2 and 7.6.5.3 .
Schema name: <schema name>	Optional; see 7.5.3 . Default: "CIM".
Schema version: <schema version>	Required; see 7.5.2 .
Schema organization: <schema organization>	Optional; see 7.5.4 . Default: "DMTF".
Central class adaptation: <central class adaptation name>	Required; see 7.6.5.4 .

Scoping class adaptation: <scoping class adaptation name>	Required; see 7.6.5.5 .
Scoping algorithm: <scoping path>	Required; see 7.6.5.6 .

<scoping path> shall be one of the following:

- If the scoping path between central class adaptation and scoping class adaptation is composed of only one association adaptation, <scoping path> shall be the name of the association adaptation.
- Otherwise, the definition of the scoping relationship shall be placed in a separate subclause of the Synopsis clause, immediately after the "Profile attributes" subclause, and be named "Scoping path". In this case <scoping path> shall have the form "see <subclause number>", where <subclause number> is the number of the scoping path subclause. In the scoping path subclause the scoping path shall be stated sequentially listing all adaptations of ordinary classes and associations that compose the scoping path, starting with the central class adaptation and ending with the scoping class adaptation.

An example of the specification of profile attributes is provided in A.2 .

9.4.5.3 Requirements for the specification of the summary

If other subclauses are defined within the Synopsis clause, the summary shall be placed in a separate subclause of the Synopsis clause that follows the "Profile attributes" subclause as required by 9.4.5.2 (or the scoping path subclause, if one is specified), and is named "Summary"; otherwise, the summary shall be placed directly within the Synopsis clause, immediately following the the profile attributes as required by 9.4.5.2 .

The first paragraph of the summary shall briefly summarize the purpose of the profile such that it may be used in other documents to describe the subject profile.

Further paragraphs may provide more detailed summary information, including text that describes the usage of the central and the scoping class.

If the subject profile is an abstract profile, the following statement shall be included as last paragraph at the end of the summary:

> "This abstract profile shall not be directly implemented; implementations shall be based on a profile that is derived from this profile."

An example of a table of class adaptations is provided in A.2 .

9.4.5.4 Requirements for the specification of references to related profiles

If other subclauses are defined within the Synopsis clause, the requirements for related profiles shall be stated in a separate subclause of the Synopsis clause that follows the Summary subclause as required by 9.4.5.3 and is named "Related profiles"; otherwise, the requirements for related profiles shall be stated directly within the Synopsis clause, immediately following the summary as required by 9.4.5.3 .

If the subject profile references other profiles, the requirements for referenced profiles shall be listed in a table of related profiles, as defined in this subclause. In that table each reference to a referenced profile shall conform to the requirements in 7.6 .

The table of related profiles shall be labeled: "Related profiles". In Table 8 requirements for columns in related profile tables are defined. Each required column is described by an entry in the list provided in Table 8 . Each list entry starts with the required name of the table column in **bold face**, followed by a dash and the requirements for cells under that column.

Table 8 – Requirements for columns of the related profiles table

Profile reference name – cell values shall state the name of the profile reference within the subject profile, as required in 7.6.2 .

Profile name – cell values shall state the registered name of the referenced profile, as required in 7.4.2 .

Organization – cell values shall state the registered organization of the referenced profile, as required in 7.4.4 .

Version – cell values shall state the value of the major and the minor version identifier of the registered version of the referenced profile that is minimally required by the subject profile, as required in 7.4.3 . The values of major and minor version identifiers shall be separated by a dot, with no interspersed whitespace characters (e.g., "1.0" or "2.14").

Relationship – cell values shall state the explicit type of profile relationship between the subject profile and the referenced profile, as required in 7.6.2 .

Description – cell values shall conform to the following rules:

- A short description of the referenced profile and its relationship to the subject profile shall be provided. The short description should focus on the use of the related profile in context of the subject profile.
- For conditional profiles the condition shall be specified using one of the mechanisms specified in 7.2.
- If the text in the Description cell exceeds twenty words, the description shall be put in a separate subclause of the Synopsis clause that is referenced from the cell.

If the subject profile does not reference other profiles, this shall be stated using the phrase "No references to other profiles are defined in this profile." instead of the table.

An example of a related profile table is provided in Annex A.2 .

9.4.5.5 Requirements for the specification of the table of features

If the subject profile defines features (see 7.12), these shall be listed in a table of features, as defined in this subclause.

NOTE Both the condensed and the traditional profile specification structure provide for the definition of features, enabling the definition of features in revisions of existing profile specifications (originally written compliance to version 1.0 of this guide) by upgrading to version 1.1 of this guide. Note though that the upgrade may require minor formal adjustments of the original version in order to become compliant with version 1.1 of this guide.

The table of features shall be labeled: "Features". In Table 14 requirements for columns in features tables are defined. Each required column is described by an entry in the list provided in Table 14 . Each list entry starts with the required name of the table column in **bold face**, followed by a dash and the requirements for cells under that column.

Table 9 – Requirements for columns of the features table

Feature name – cell values shall state the name of the feature, see 7.12.3 .

Granularity – cell values shall state whether the feature can be implemented for the profile as a whole, or for specific adaptation instances.

The following rules apply:

- If the feature can be implemented for the profile as a whole, the Granularity cell value shall be "profile".
- If the feature can be implemented for specific adaptation instances, the Granularity cell value shall be the name of the adaptation, followed by "instance".

Requirement – cell values shall state the requirement level of the feature

The following rules apply:

- If the feature is conditional, the cell value shall be "Conditional".
- If the feature is optional, the cell value shall be "Optional".

Description – cell values shall provide a description of the feature.

– The feature definition subclause in the Implementation clause (see 9.4.7.3) shall be referenced. No other text should be added.

If the specified profile does not define features, the text "No features are defined in this profile." should be stated instead of the table.

An example of a table of class features is provided in A.2 .

9.4.5.6 Requirements for the specification of the table of class adaptations

The class adaptations (see 7.10) defined in the subject profile shall be listed in a table of class adaptations, as defined in this subclause.

The placement of the table depends on the profile specification structure that is applied by the subject profile, as follows.

If the traditional profile specification structure is applied by the subject profile,, the table of class adaptations shall be specified in the Overview subclause of the "CIM elements" clause (see 9.4.10.2); in this case the provisions in the remaining part of this subclause do not apply.

If the condensed profile specification structure is applied by the subject profile, the provisions in the remaining part of this subclause apply.

The table of class adaptations shall be specified as part of the Synopsis clause. All class adaptations (including the adaptations of ordinary classes, of associations and of indications) defined by the subject profile shall be listed in the table of class adaptations.

The table of class adaptations shall be labeled: "Class adaptations". In Table 14 requirements for columns in class adaptation tables are defined. Each required column is described by an entry in the list provided in Table 14 . Each list entry starts with the required name of the table column in **bold face**, followed by a dash and the requirements for cells under that column.

Table 10 – Requirements for columns of the class adaptations table

Adaptation – cell values shall state the name of the adaptation, see 7.10.2.1 .

– If an adaptation is based on other class adaptations, the cell in the Adaptation column shall span all the cells in the other columns that are related to the specified adaptation.

Elements – cells may be split into subcells, as follows:

– The first subcell shall contain the name of the adapted class.

– If base adaptations are defined, these shall be stated in subsequent subcells, using the following ABNF defined format:

```
AdaptationReference = ProfileName "::" AdaptationName
```

The value of `ProfileName` shall be the registered name (see 7.4.2) of the referenced profile that defines the referenced adaptation, and the value of `AdaptationName` shall be the name of the referenced adaptation, as defined by its defining profile.

– If a standard message is defined for an indication adaptation, that shall be stated in a subsequent subcell.

Requirement – cell values shall state the requirement level for the adaptation, as defined in 9.2.4.3.

- If an adaptation is based on other class adaptations, and different requirement levels apply, these shall be specified in separate cells in this column; however, within the scope of a cell in the Adaptation column, if all corresponding cells in the Elements column are required with the same requirement level, the respective subcells in the Requirement column may be collapsed into one cell containing the common requirement level.

Description – cell values shall provide a description of the adaptation.

The following rules apply:

- If the requirement level is conditional, and unless the condition is already stated in the Requirement column, the condition shall be stated here, as detailed in 9.2.4 .
- a textual description shall be provided that describes the purpose of the adaptation. The description should describe the managed object type that is modeled by the adaptation, unless that is already addressed with sufficient precision by the schema descriptions of the adapted class.
- For trivial class adaptations defined by the subject profile that do not specify additional requirements beyond those defined in the schema definition of the adapted class:

  `"See CIM schema definition."`
- If the corresponding cell in the Elements column is split into subcells, the cell in the Description column shall be split into respective subcells, unless the description applies in all cases.
- If the value in any Description subcell exceeds twenty words, a separate adaptation definition subclause shall be provided within the Implementation clause; for details, see 9.4.7.4.3. In this case the description shall be provided as part of the adaptation definition subclause, and the adaptation definition subclause shall be referenced from the table entry, as follows:

  `"See" AdaptationSubclauseNumber "."`

  `AdaptationSubclauseNumber` is the number of the adaptation definition subclause.

An example of a table of class adaptations is provided in A.2 .

9.4.6 Requirements for the specification of the Description clause

This subclause defines requirements for the Description clause in profile specifications.

Each profile specification shall have a Description clause.

The Description clause in profile specifications

- shall provide an overview of the subject profile.
- should describe the management domain addressed by the subject profile, and the major object types for that the subject profile defines adaptation.
- shall not include normative definitions.
- should contain diagrams that detail the purpose of the subject profile.
  - The Description clause of profile specifications written in conformance with the condensed structure (see 9.3.2) should contain one or more DMTF collaboration structure diagrams (see 9.2.2.3) that detail the collaboration defined by the subject profile. Each adaptation defined by the subject profile shall appear at least once.
  - The Description clause of profile specifications written in conformance with the traditional structure (see 9.3.3) should contain one or more DMTF class diagrams (see 9.2.2.4) that detail the model defined by the subject profile.

– In addition, the Description clause may contain DMTF object diagrams (see 9.2.2.5) providing details on CIM instances, their interactions and their relationship to managed objects in managed environments, as required by the subject profile.

An example of a Description clause is provided in A.3 .

9.4.7 Requirements for the specification of the Implementation clause

This subclause defines requirements for the Implementation clause in profile specifications.

9.4.7.1 General

Each profile specification shall have an Implementation clause.

If the profile is a derived profile that does not add specifications for implementations beyond those defined in its base profile(s), the Implementation clause shall only contain the statement “All implementation requirements are defined in base profile(s)”.

9.4.7.2 Usage of subclauses

The Implementation clause should be structured into subclauses.

Subclauses may introduce subtopics that apply to one or more profile elements, such as for example a subclause titled "Element discovery", or may introduce subtopics that address specific profile elements such as for example a specific class adaptation defined in a subclause titled "Adaptation: Fan: CIM_Fan".

Subclauses of the Implementation clause should be ordered as follows:

- Subclauses that describe the management domain and managed object types
- Subclauses that introduce concepts
- An optional Features subclause, as detailed in 9.4.7.3
- A required Adaptations subclause, as detailed in 9.4.7.4 .

NOTE ISO/IEC Directives, Part2 require that at each subclause level at least two subclauses are specified. For that reason, in the case where according to this guide only the Adaptations subclause would be required, ISO/IEC Directives, Part2 would require another subclause of the Implementation clause, and recommend a subclause named "General" with a brief introduction.

9.4.7.3 Requirements for the specification of features

If the subject profile defines features (see 7.12), the Implementation clause shall contain a separate subclause named "Features".

The Features subclause of the Implementation clause shall contain a separate subclause for each defined feature.

The title of each feature specific subclauses shall be formatted as follows:

```
FeatureSubclauseTitle = "Feature: " FeatureName
```

The value of `FeatureName` shall be the name of the feature; see 7.12.3.

If the feature is conditional, that shall be stated first in the feature definition subclause, along with the specification of the condition, following the conventions established in 9.2.4 .

Each feature definition subclause shall provide all of the following:

- A description of the feature as defined in 7.12 . The description may be provided directly within the feature definition subclause, or within a separate subclause titled "Feature description".
- A description of one or more discovery mechanisms for the feature. The discovery mechanisms may be described directly within the feature definition subclause, of within a separate subclause

titled "Feature discovery". See 7.12.6 for requirements for the definition of discovery mechanisms.

The implementation requirements that result from a decision to implement a feature are not defined as part of the feature definition subclause. Instead these requirements are specified as conditional requirements for other profile definitions such as related profiles, class adaptations, or - within the definition of class adaptations – references to and additional requirements for properties or methods. The condition in these cases is a feature implementation condition, such that respective requirements apply in case the feature is implemented. This approach enables the specification of profile elements that depend on more than one feature.

9.4.7.4 Requirements for the specification of adaptations

This subclause defines requirements for the specification of adaptations.

9.4.7.4.1 General

The Implementation clause shall contain a separate subclause named "Adaptations".

The Adaptations subclause of the Implementation clause shall contain a separate subclause for each class adaptation (including adaptations of associations or indications) defined by the profile as specified in 9.4.7.4.3, unless the class adaptation is a trivial class adaptation.

A trivial class adaptation does not define additional requirements beyond those defined by the adapted class and its base adaptations. Trivial class adaptations typically are defined as a point of reference for other profiles, such that referencing profiles can define class adaptations based on them. The description of a trivial class adaptation may be solely provided in the entry in the table of class adaptations within the Synopsis clause if the space requirements for table cells are met; see 9.4.5.6 .

9.4.7.4.2 Requirements for the specification of operation requirements

The Adaptation subclause of the Implementation subclause shall contain a subclause named "Profile conventions for operations". That subclause shall contain one or more lists of operations that are referred to by the definition of individual class adaptations.

The "Profile conventions for operations" subclause shall contain the text:

"For each class adaptation (including those of associations), the implementation requirements for operations, including for those in the following default list, are specified in class-adaptation-specific subclauses of the Adaptations subclause."

A profile may define a default list of operations for adaptations of ordinary classes and associations. A profile shall only list operations that are defined in the operations specification referenced by the profile. The default list of operations shall be stated as follows:

"The default list of operations for all class adaptations is as follows:

operation-1

operation-2

…"

The default list may be extended for adaptations of associations, as follows:

"For association adaptations, the default list of operations includes the following operations in addition:

a-operation-1

a-operation-2

…"

The applicability of the default list shall be specified in class specific subclauses within the Adaptations subclause of the Implementation clause; see 9.4.7.4.6 .

9.4.7.4.3 Requirements for the specification of individual class adaptations

Each class adaptation definition subclause within the Adaptation subclause of the Implementation clause shall be titled

```
AdaptationClauseTitle = "Adaptation:" AdaptationName ":" AdaptedClassName
```

`AdaptationName` is the name of the adaptation (see 7.10.2), and `AdaptedClassName` is the name of the adapted class.

The class adaptation subclause may contain a first subclause titled "General" that defines general requirements of the class adaptation that do not fit into any of the other subclauses.

Each adaptation subclause shall define implementation requirements. Implementation requirements may be defined directly within the subclause that containing the definition of the class adaptation, or within a separate subclause titled "Implementation requirements".

Implementation requirements for properties and methods shall be stated in the form of a table that references those properties and methods of the adapted class that are assigned a requirement level by the class adaptation, and optionally are further constrained; if required, the table entries may refer to other subclauses that provide detail information.

The table listing elements of the adaptation shall be labeled:

```
AdaptationElementTableTitle = AdaptationName ":" AdaptedClassName
```

`AdaptationName` is the name of the adaptation (see 7.10.2), and `AdaptedClassName` is the name of the adapted class.

In Table 11 requirements for columns in adaptation element tables are defined. Each required column is described by an entry in the list provided in Table 11 . Each list entry starts with the required name of the table column in **bold face**, followed by a dash and the requirements for cells under that column.

Table 11 – Requirements for columns of adaptation element tables

Element – cell values shall state property requirements (see 7.10.2.5) or method requirements (see 7.10.3.1) defined by the adaptation. Method names shall be suffixed with "( )". **Requirement** – cell values shall state the requirement level of the property requirement or method requirement. The requirement level shall be stated in conformance to the conventions defined in 9.2.3 . **Description** – cell values shall conform to the following specifications: – If the requirement level is "conditional", and unless the condition is already stated in the Requirement column, the condition shall be stated here, as detailed in 9.2.4 . – The definition of additional requirements. This may include a combination of any of the following elements: For property requirement on all types of CIM properties (including references): – the definition of the attribute or characteristic of the object type in the management domain that is represented by the CIM property – the definition of property value patterns for string properties – the definition of value constraints – the definition of default values

– the keyword "Deprecated" if the property is marked deprecated by the profile, or in the schema definition; for details, see 7.16 .

For property requirement on CIM references:

the definition of the multiplicity as detailed in 9.2.5.2.

For method requirements:

– the definition of additional semantics beyond those defined in the schema, with particular attention on the effects in the managed environment

– the keyword "Deprecated" if the method is marked deprecated by the profile, or in the schema definition; for details, see 7.16 .

- The cell value should contain not more than twenty words. Text longer than twenty words should be placed in a separate subclause of the adaptation specific subclause, and referenced from the table cell. The requirements for such separate subclauses are detailed in 9.4.7.4.4 (referenced properties) and 9.4.7.4.5 (referenced methods).

NOTE Version 1.0 of this guide defined "Notes" as the title of the third column; this was changed to "Description" for coherent definition of tables specified in this guide. Many profiles based on version 1.0 of this guide use "Description" already.

The implementation of key properties and properties marked by the Required qualifier in the schema definition of the adapted class is always required; for details, see 8.1.2. In order to avoid the replication of implementation requirements, such properties shall be only be included in the table if the subject profile imposes additional constraints; otherwise, such properties should not be listed.

Optional properties and methods shall not be listed unless the profile defines additional requirements for these elements beyond those defined in the schema.

9.4.7.4.4 Requirements for the specification of property requirements

The title of property specific subclauses within subclauses defining a class adaptation shall be formatted as follows:

```
PropertySubclauseTitle = "Property: " PropertyName ( "[ ]" )
```

The property specific subclause shall specify a relationship to the aspect of managed objects represented by adaptation instances that is reflected by the property as required in 7.10.2.5 .

If value constraints are defined, the conventions defined in 9.2.5 shall be applied.

9.4.7.4.5 Requirements for the specification of method requirements

The title of method specific subclauses within subclauses defining a class adaptation shall be formatted as follows:

```
MethodSubclauseTitle = "Method: " MethodName "( )"
```

The method specific subclause should provide a description detailing the semantics of the method as required in 7.10.3.1. The description may contain references to use cases (see 9.4.9).

The description of the method parameters required by the subject profile shall be provided in a table.

The table shall be labeled:

```
ParameterTableTitle = MethodName "( ): Parameters"
```

In Table 12 requirements for columns in method parameter tables are defined. Each required column is described by an entry in the list provided in Table 12 . Each list entry starts with the required name of the table column in **bold face**, followed by a dash and the requirements for cells under that column.

3124 **Table 12 – Requirements for columns in method parameter tables**

Qualifiers – cell values shall state parameter qualifiers as follows: – the cell value shall list the textual value "In" if and only if the effective value of the In qualifier for the parameter is true. – the cell value shall list the textual value "Out" if and only if the effective value of the Out qualifier for the parameter is true. – the cell value shall list the textual value "Req" if and only if the effective value of the Required qualifier for the parameter is true. – a profile specification shall not change the interpretation of the value of the schema defined In, Out and Required qualifiers, just present their effective values. NOTE The textual value "Req" in a cell under the Qualifiers column does not indicate whether or not the profile requires an implementation of the parameter. – multiple textual values shall be separated by commas
Name – cell values shall state the parameter name
Type – cell values shall state the parameter type
Description/Values – cell values shall provide details about the use of the parameter as required by the profile. – If value constraints are defined, the conventions defined in 9.2.5 shall be applied. – The value in a Description/Value table cell should contain not more than twenty words. Text longer than twenty words should be placed in a subclause of the method specific subclause and referenced from the table cell.

3125 If the schema descriptions of method parameters adequately describe the use of the method parameters
3126 as required by the subject profile, then the method specific subclause shall refer to the method parameter
3127 description in the schema with a statement "See schema description.".

EDITORIAL NOTE: We are considering adding a "Requirement level" column to the method parameter table in order to make it more similar to the property requirements table.

The Req qualifier is not suitable here, because it only reflects a schema setting, but does not allow a profile to specify with which requirement level a particular parameter is required.

3128 If the schema descriptions of method return values does not adequately describe their use as required by
3129 the subject profile, the method specific subclause shall provide a table specifying return values.

3130 The table shall be labeled:

3131 `ReturnValueTableTitle = MethodName "( ): Return values".`

3132 In Table 13 requirements for columns of the return value table are defined. Each column is described by
3133 an entry in the list provided in Table 13 . Each list entry starts with the required name of the table column
3134 in **bold face**, followed by a dash and the requirements for each cell within that column.

3135 **Table 13 – Requirements for columns of the return value table**

Value – cell values shall state the numeric return value followed by the corresponding string description in parenthesis. For example: "1 (Not Implemented)".
Description – cell values shall provide details about the situation indicated by the return value. – If a return value only applies under certain conditions, this shall be stated in the form "Applica-

ble only if the <conditional element> is implemented.".

– The value in a Description table cell should contain not more than twenty words. Text longer than twenty words should be placed in a subclause of the method specific subclause and referenced from the table cell.

If the schema descriptions of method return values adequately describe their use as required by the subject profile, the method specific subclause should refer to the schema. For example, a Fan profile describing return values for the RequestStateChange( ) method applied to instances of the CIM_Fan class representing fans might state "For return values see the schema definition of the CIM_EnabledLogicalElement class.

If the subject profile specifies the use of standard messages for a method, these shall be stated as defined in 9.4.7.4.8 . If the subject profile does not specify use of standard messages for a method, no table shall be provided in the method specific subclause; instead, the method specific subclause shall contain the statement "No standard messages are defined.".

9.4.7.4.6 Requirements for the specification of operations requirements

Each adaptation definition subclause that defines an adaptation of an ordinary class or of an association shall state operations requirements. A profile shall specify requirements only for operations that are defined in the operations specification referenced by the profile.

Subsequent definitions in this subclause make use of the following ABNF rules:

- `TableNum` shall be the number of the table
- `OpSpec` shall be a reference to the operations specification
- `PcoNum` shall be the subclause number of the "Profile conventions for operations" subclause
- `OperationName` shall be the name of an operation for that requirements are defined by the subject profile.

If a default list of operations is defined in the "Profile conventions for operations" subclause (see 9.4.7.4.2), and the default list shall apply unmodified for the specified class adaptation, the following statement (including the NOTE) shall be provided:

"All operations in the default list in" `PcoNum` "shall be implemented as defined in" `OpSpec` "."

"NOTE Base adaptations may define additional requirements on operations."

If a default list of operations is defined, and if additional operations are specified for the specified class, a table shall be provided that details implementation requirements for specific operations that are not covered by the default requirement.

The table shall be preceded by the statement (including the NOTE):

"Table" `TableNum` "lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in" `OpSpec` ". In addition, and unless otherwise stated in Table" `TableNum` ", all operations in the default list in" `PcoNum` "shall be implemented as defined in" `OpSpec` "."

"NOTE Base adaptations may define additional requirements on operations."

If a default list of operations is not defined, a table shall be provided that lists each operation that is required to be implemented by the specified profile. The table shall be preceded by the statement (including the NOTE):

"Table" `TableNum` "lists implementation requirements for operations. If implemented, these shall be implemented as defined in" `OpSpec` "."

"NOTE Base adaptations may define additional requirements on operations."

If a table is provided, it shall be labeled as follows:

```
OperationsTableTitle = AdaptationName "operations"
```

In Table 14 requirements for columns of the operations table are defined. Each column is described by an entry in the list provided in Table 14 . Each list entry starts with the required name of the table column in **bold face**, followed by a dash and the requirements for each cell within that column.

Table 14 – Requirements for columns of the operations table

Operation – cell values shall state the name of the operation, as follows:

```
OperationCellValue = OperationName "( )"
```

Requirement – cell values shall state the requirement level for the operation (see 9.2.3)

Description – cell values may contain a reference to an operation specific subclause and/or refer to a table of standard messages. In this case the referenced table shall be formatted as specified in 9.4.7.4.8.

- If for a particular operation the use of messages by the subject profile is within the set of messages that the referenced operations specification defines for that operation, no message descriptions are required in the subject profile . In this case a statement "For message descriptions, see `OpSpec`." should refer to the operations specification.

If the profile defines behavior for selected operations that extends the behavior defined in the referenced operations specification (for example by requiring a specific effect on the managed environment), these operations shall be documented in separate subclauses of the subclause that contains the definition of the class adaptation. These subclauses shall be titled as follows:

```
OperationSubclauseTitle = OperationName "( )"
```

The subclauses shall describe the requirements for the operation – including all side effects and expected results in the managed environment. These subclauses shall be referenced from the Requirements column for the appropriate operation in the table defined above.

9.4.7.4.7 Requirements for the specification of operations related to associations

Operations that result in instances (or instance names) relative to a source instance are called association traversal operations. Association traversal operations shall be specified in the operations subclause for the adaptation of the source class, not in the operations subclause of the association adaptation. In addition, a particular source adaptation defined by the subject profile can be the starting point for the traversal of more than one association adaptation. If in this case the operation requirements are different for each association that can be traversed, a separate operations specifications are required within operation requirements of the source class for each traversable association.

For example, if a profile defines operations as defined in DSP0223 in order to traverse the CIM_SystemDevice association, the GetAssociatedInstances( ) and GetAssociatedInstancePaths( ) operations would not be listed in the intrinsic operations table of the adaptation of the CIM_SystemDevice association; instead, the intrinsic operations tables for adaptations of the classes that are referenced by the adapted association, such as in this case adaptations of the CIM_System class and the CIM_LogicalDevice class would list respective operations.

NOTE Associations may be adapted such that subclasses of the classes referenced by the association apply; see 7.10.2.5.

9.4.7.4.8 Requirements for the specification of requirements related to standard messages

If the subject profile specifies the use of standard messages for a method or operation, the specifications in this subclause apply.

Requirements for standard messages shall be defined in a table.

The table shall be labeled as follows:

```
StandardMessageTableTitle = ActivityName "( ) standard messages"

ActivityName = MethodName | OperationName
```

In Table 15 requirements for columns of the standard message table are defined. Each column is described by an entry in the list provided in Table 15 . Each list entry starts with the required name of the table column in **bold face**, followed by a dash and the requirements for each cell within that column.

Table 15 – Requirements for columns of the standard message table

(return) Message ID – cell values shall state a return value in parenthesis followed by the name of the registering organization and the message ID from that organization
Message – cell values shall state the message text (abbreviated, if appropriate).

Each table cell should contain not more than twenty words. If more than twenty words are required, respective content shall be place in a separate subclause and referenced from the table.

9.4.7.4.9 Requirements for the specification of instance requirements

Each adaptation definition subclause that defines an adaptation of an ordinary class or of an association class shall state instance requirements, as defined in 7.10.3.3. Instance requirements may be specified as part of the implementation requirements, or may be specified in a separate subclause.

9.4.7.4.10 Requirements for the specification of indication generation requirements

Each adaptation definition subclause that defines an adaptation of an indication shall state indication generation requirements, as defined in 7.10.4.3. Indication generation requirements may be specified as part of the implementation requirements, or may be specified in a separate subclause.

Deprecated content - start

9.4.8 Requirements for the specification of the Methods clause

This subclause details requirements for the Methods clause in profile specifications.

9.4.8.1 General

Each profile specification that applies the traditional profile specification structure (see 9.3.3) shall contain a Methods clause. Profile specifications that apply the condensed profile specification structure shall not contain a Methods clause because in this case respective content is specified in the Implementation clause already.

9.4.8.2 Requirements for the specification of methods

This subclause specifies the definition of the adaptation of methods in profile specifications.

9.4.8.2.1 General

The Methods clause shall contain an "Extrinsic methods" subclause.

If the profile specification specifies a specialized profile that does not add requirements for methods, but one or more of its base profile(s) defines requirements for methods, the "Extrinsic methods" subclause shall only contain the statement "All method requirements are defined in base profile(s).".

If the profile specification specifies a profile that does not add adaptations for extrinsic methods, the "Extrinsic methods" subclause shall only contain the statement "No method requirements are defined.".

9.4.8.2.2 Method specific subclauses

Each extrinsic method the is referenced by a class adaptation defined subject profile shall be specified in a separate subclause of the "Extrinsic methods" subclause.

The title of method specific subclauses shall be formatted as follows:

```
MethodSubclauseTitle = ClassAdaptationName "." MethodName "( )"
```

`ClassAdaptationName` shall be the name of the class adaptation, `MethodName` shall be the name of the method.

Method specific subclauses shall be referenced from the subclause of the CIM elements clause that defines the class adaptation referencing the method; see 9.4.10.3 .

Method specific subclauses shall conform to the requirements of 9.4.7.4.5 .

9.4.8.3 Requirements for the specification of the Operations subclause

This subclause details requirements for the Operations subclause of the Methods clause in profile specifications.

9.4.8.3.1 General

The Methods clause shall contain a "Generic operations" subclause.

If the profile specification specifies a specialized profile that does not add requirements for operations, the "Generic operations" subclause shall only contain the statement "All operation requirements are defined in base profile(s).".

9.4.8.3.2 Requirements for the specification of the "Profile conventions for operations" subclause

The "Generic operations" subclause shall contain a "Profile conventions for operations" subclause unless the profile is a specialized profile that does not add specifications for operations beyond those defined in its base profile(s).

The "Profile conventions for operations" subclause shall specify conventions applied by the profile for the specification of requirements for operations; it shall follow the method specific subclauses (if any).

The "Profile conventions for operations subclause" shall state the operations specification that rules the definition of operations in the profile as required in 7.10.3.2 . For example, "This profile defines operations in terms of DSP0223".

Table 16 defines three options one of which shall be applied by a profile specification for the "Generic operations" subclause:

Table 16 – Profile convention options

Option	Requirements for the Intrinsic operations subclause
Option 1 – table includes each operation for each	**Deprecated** with version 1.0.1; now covered by option 2, with additional requirements specified in 9.4.8.3.3 .

class	"Support for operations for each profile class (including associations) is specified in the following subclauses. Each of these subclauses includes a table listing all the operations supported by this profile. Compliant implementations of this profile shall support all these operations."
Option 2 – table includes operations with profile-specific requirements. The operations in the default list apply to the extent detailed in adaptation specific subclauses of the Methods clause.	The "Profile conventions for operations" subclause of the Methods clause shall contain the text: "For each adaptation (including association adaptations), the implementation requirements for operations, including for those in the following default list, are specified in adaptation specific subclauses of `OpScNumber` ." `OpScNumber` is the number of the Operations subclause of the Methods clause. A profile may define a default list of operations, as follows: "The default list of operations is as follows: operation-1 operation-2 …" The default list may be extended for adaptations of classes referenced by an association, as follows: "For adaptations that are referenced by an association adaptation, the default list of operations includes the following operations in addition: a-operation-1 a-operation-2 …" The applicability of the default list shall be specified in adaptation specific subclauses of the Operations subclause of the Methods clause; see 9.4.8.3.3 .
Option 3 – table includes operations with profile-specific requirements. Other operations may be implemented.	**Deprecated** with version 1.0.1; now covered by option 2, with additional requirements specified in 9.4.8.3.3 . "Support for operations for each profile class (including associations) is specified in the-following subclauses. Each of these subclauses includes either • a statement “All operations from the default list specified in section nnn are supported as described by DSPXXXX vX.y.z” where nnn is the number of the section containing the default list. • a table listing all the operations that are not constrained by this profile or where the profile requires behavior other than described by DSPXXX. The default list of operations is operation-1, operation-2, … Profile requirements for these operations are specified in the "Requirements" column .

The default list of intrinsic operations for ordinary classes typically lists the intrinsic operations related to manipulation of instances and possibly intrinsic operations to execute queries.

9.4.8.3.3 Requirements for the specification of class specific operations subclauses

A subclause shall be included for each class adaptation (including association adapations) defined by the subject profile. The requirements of 9.4.7.4.6 apply.

For operations related to associations the requirements of 9.4.7.4.7 apply.

Deprecated content - end

9.4.9 Requirements for the specification of the Use-cases clause

This subclause details requirements for the Use-cases clause in profile specifications.

9.4.9.1 General

Each use-case shall be documented in a separate subclause.

9.4.9.2 Requirements for the specification of subclauses containing object diagrams

A profile specification may contain zero or more subclauses with object diagrams depicting typical situations that a client may observe in the process of applying use-cases defined by the profile. Each object diagram subclause shall contain one object diagram; the rules defined in 9.2.2.5 apply.

The title of object diagram subclauses shall be formatted as follows:

```
ObjectDiagramSubclauseTitle = SituationName
```

`SituationName` shall state a name that enables a human reader to grasp the situation that the object diagram depicts; the name shall be unique within the profile specification.

A brief description of the object diagram should be provided, with particular attention on the managed objects in the managed environment and their relationships that are represented by the CIM instances depicted in the object diagram.

9.4.9.3 Requirements for the specification of use-case specific subclauses

9.4.9.3.1 General

Each use-case shall be specified in a separate subclause of the Use-cases clause of a profile specification.

The title of use-case specific subclauses shall be formatted as follows:

```
UseCaseSubclauseTitle = UseCaseName
```

`UseCaseName` shall state a name for the use-case that enables a human reader to grasp the intent of the use-case; the name shall be unique within the profile.

Each use-case specific subclause should contain a brief description of the use-case.

See Appendix A.5 for examples of use-cases.

9.4.9.3.2 Requirements for the specification of preconditions in use-cases

The definition of preconditions as required by 7.13.2 shall be provided within a first subclause within any the use-case specific subclause. The precondition subclause shall be titled "Preconditions". If there is more than one precondition, a list format should be used.

The situation as described by the preconditions should be presented in the form of an DMTF object diagram; DMTF object diagrams may be shared by multiple use-cases, as detailed in 9.4.9.2 .

9.4.9.3.3 Requirements for the specification of flows of activities in use-cases

The description of flows of activities as required by 7.13.3 shall be provided in a separate subclause within any use-case specific subclause. The following formal requirements apply:

- Descriptions may contain references to DMTF object diagrams .
- Normative requirements shall not be duplicated in the use-cases.
- Parameters should be stated in a list format where each list entry describes one parameter. If a parameter is an embedded CIM instance, a list format should be used to state names and values of required or applicable properties. Descriptions of parameters or properties should provide an interpretation of their use in the management domain.
- The inspection of method results and return parameters may be described either as part of a use-case step after the description of a method invocation, or as separate use-case steps.
- The main flow of activities should be the sequential processing of use-case steps; however, the following phrases may be used to indicate deviations:

- "<step post condition>; the use-case continues with step <step number>.", where <step post condition> details a simple post condition of the use-case step such as a return value and its significance. If more than one next step is possible, each should be listed.
- "This completes the use-case; the postconditions in <subclause number> apply."; this phrase describes a normal completion of the use-case.
- "This terminates the use-case; the postconditions in <subclause number> apply."

Alternatively to the format defined above, use-cases may be presented as pseudo-code.

9.4.9.3.4 Requirements for the specification of postconditions in use-cases

The definition of a postconditions as required by 7.13.4 shall be provided in a separate subclause within the use-case specific subclause that is titled "Postconditions".

Postcondition subclauses may be further subdivided into subclauses, addressing various situation resulting from processing the use-case such as for example success or failure. Such situations may likewise be presented by other structuring elements such as lists; however, if separate subclauses are required in the case where referenced for example from the description of an activity flow.

Deprecated content - start

9.4.10 Requirements for the specification of the "CIM elements" clause

This subclause details requirements for the "CIM elements" clause in profile specifications.

9.4.10.1 General

Each profile specification that applies the traditional profile specification structure (see 9.3.3) shall contain a "CIM elements" clause. Profile specifications that apply the condensed profile specification structure shall not contain a "CIM elements" clause because in this case respective content is specified in the Implementation clause already.

Version 1.0 of this guide did not formally define the concept of adaptations; instead it informally used the terms "class", "profile class" or "supported class". Revisions of existing profile specifications that apply this version of this guide (version 1.1.0_wgv0.7.17) should start using the term adaptation in modified text passages; however, it is not required to modify otherwise unmodified text solely for the introduction of these new terms. The use of these terms in this guide shall apply correspondingly to entities such as "class", "profile class" or "supported class" as used by version 1.0 of this guide.

If the subject profile is a derived profile that does not add specifications for "CIM elements" beyond those defined in its base profile(s), the "CIM elements" clause shall contain the statement "All CIM elements are defined in base profile(s)".

NOTE Typical examples of derived profiles not adding specifications for CIM elements are those derived from an abstract profile for the sole purpose of providing a base for an implementation. Recall that abstract profiles must not be implemented directly.

The "CIM elements" clause shall contain the following subclauses:

- An initial Overview subclause; see 9.4.10.2 .
- A subclause for each adaptation defined by the profile; see 9.4.10.3 .

9.4.10.2 Requirements for the specification of the Overview subclause

This subclause details requirements for the Overview subclause of the "CIM elements" clause.

The Overview subclause shall contain a table listing the adaptations defined by the profile (including associations and indications). The table shall be labeled:

```
CIMElementTableTitle = ProfileName "profile : CIM elements"
```

3366 `ProfileName` shall be the registered name of the profile. Each entry in the table shall declare an
3367 adaptation defined by the subject profile.

3368 The table shall have four columns:

AdaptationName – cell values shall state the name of the adaptation; see 7.10.2.1 .

Elements – cells may be split into subcells, as follows:

- The first subcell shall contain the name of the adapted class.
- If base adaptations are defined, these shall be stated in subsequent subcells, using the following ABNF defined format:

  ```
  AdaptationReference = ProfileName "::" AdaptationName
  ```

  The value of `ProfileName` shall be the registered name (see 7.4.2) of the referenced profile that defines the referenced adaptation, and the value of `AdaptationName` shall be the name of the referenced adaptation, as defined by its defining profile.
- If a standard message is defined for an indication adaptation, that shall be stated in a subsequent subcell.

Requirement – cell values shall state the requirement level for the adaptation, as defined in 9.2.4.3 .

- If an adaptation is based on other adaptations, and different requirement levels apply, these shall be specified in separate subcells in this column; however, within the scope of a cell in the Adaptation column, if all corresponding cells in the Elements column are required with the same requirement level, the respective subcells in the Requirement column may be collapsed into one cell containing the common requirement level.

Description – cell values shall contain a description of the adaptation.

The following rules apply:

- If the requirement level is "conditional", and unless the condition is already stated in the Requirement column, the condition shall be stated here, as detailed in 9.2.4 .
- a textual description shall be provided that describes the purpose of the adaptation. The description should describe the managed object type that is modeled by the adaptation, unless that is already addressed with sufficient precision by the schema descriptions of the adapted class.
- For trivial class adaptations defined by the subject profile that do not specify additional requirements beyond those defined in the schema definition of the adapted class, that shall be indicated by the following statement:

  ```
  "See CIM schema definition."
  ```
- If the corresponding cell in the Elements column is split into subcells, the cell in the Description column shall be split into respective subcells, unless the description applies in all cases, inwhich case respective subcells in the Description column may be collapsed into one cell containing the common description.
- If the value in any Description subcell exceeds twenty words, a separate adaptation definition subclause shall be provided within the Implementation clause; for details, see 9.4.7.4.3. In this case the description shall be provided as part of the adaptation definition subclause, and the adaptation definition subclause shall be referenced from the cell, as follows:

  ```
  "See" AdaptationSubclauseNumber "."
  ```

  `AdaptationSubclauseNumber` is the number of the subclause of the Implementation clause that contains the definition of the adaptation.

3369

3370 **9.4.10.3 Requirements for the specification of subclauses defining class adaptations**

3371 The specification of the each class adaptation subclause shall be in compliance with 9.4.7.4, with the
3372 following admissible deviations:

3373 • The title of the subclause may apply the deprecated naming convention using the name of the
3374 adapted class and a modifier; for details see 7.10.2.1 .

3375 **Deprecated content – end**

ANNEX A (Informative) Examples

A.1 General

All the examples provided within ANNEX A provide excerpts from a hypothetic Example Fan profile. The examples are related to each other, but together would not form a complete profile specification.

A.2 Example of a Synopsis clause

Table 17 provides an example of a Synopsis clause; see 9.4.5 for requirements on the specification of the Synopsis clause.

Table 17 – Example of Synopsis clause

X-5 Synopsis

Profile name: Example Fan

Version: 1.1.0

Organization: DMTF

Schema version: 2.24

Profile type: Component

Central class adaptation: Fan

Scoping class adaptation: ComputerSystem

Scoping algorithm: FanInSystem

The Example Fan profile extends the management capability of a scoping profile by adding the capability to describe fans and redundant fans within managed systems.

Table X-1 list profiles that are referenced by this profile.

Table X-1 – Related profiles

Profile reference name	Profile name	Organi-zation	Version	Relationship	Description
FanProfileRegistra-tion	Profile Registration	DMTF	1.1	Mandatory	The Profile Registraion profile applied for the registration of implemen-tations of the Example Fan profile.
FanPhysicalAsset	Physical Asset	DMTF	1.1	Optional	The Physical Asset profile applied for fans as physical assets.
FanSensors	Sensors	DMTF	1.1	Conditional	The Sensors profile applied for sensors of fans. Condition: The FanSpeedSensor feature is implemented; see X-7.2.3 for feature defini-tion.

Table X-2 list the features defined in this profile.

Table X-2 – Features

Feature name	Granularity	Requirement	Description
FanStateManagement	Fan instance	Optional	See X-7.2.1 for feature definition
FanElementNameModification	Fan instance	Optional	See x.x for feature definition
FanSpeedSensor	Fan instance	Conditional	See X-7.2.3 for feature definition
FanLifecycleIndications	Profile	Conditional	See X-7.2.4 for feature definition
FanProcessIndicationsForLifecycleEvents	Profile	Optional	See X-7.2.5 for feature definition

Table X-3 lists the class adaptations defined in this profile.

Table X-3 – Class adaptations

Adaptation	Adapted class and base adaptations	Requirement	Description
Classes			
Fan	CIM_Fan	Mandatory	See X-7.4.3 .
	Profile Registration::CentralElement	Mandatory	
	Example Sensors::Sensor	Conditional	
FanInSystem	CIM_SystemDevice	Mandatory	See X-7.4.4 .
FanCapabilities	CIM_EnabledLogicalElementCapabilities	Conditional	See X-7.4.5 .
CapabilitiesOfFan	CIM_ElementCapabilities	Conditional	See X-7.4.6 .
CooledElement	CIM_ManagedElement	Mandatory	See
…	…	…	…
FanSensor	CIM_Sensor	Conditional	See X-7.4.7 .
	Example Sensors::Sensor		
FanNumericSensor	CIM_NumericSensor	Conditional	See X-7.4.8 .
	Example Sensors::NumericSensor		
SensorOfFan	CIM_AssociatedSensor	Conditional	See X-7.4.9 .
	Example Sensors::AssociatedSensor		
…	…	…	…
FanImplementation	CIM_RegisteredProfile	Mandatory	See
	Profile Registration::Implementation		
SensorsForFan	CIM_ReferencedProfile	Mandatory	None
	Profile Registration::ReferencedProfile		
…	…	…	…
System	CIM_System	Mandatory	Scoping class adaptation; scoping profiles base their central class adaptation on this adaptation.
	Profile Registration::ScopingElement		

…	…	…	…
FilterCollection	CIM_FilterCollection	Conditional	See X-7.4.20 .
	Indications::FilterCollection		
IndicationFilter	CIM_IndicationFilter	Conditional	See X-7.4.21 .
	Indications::IndicationFilter		
MemberOfFilter-Collection	CIM_MemberOfCollection	Conditional	See X-7.4.22 .
	Indications::MemberOfCollection		
…	…	…	…
Indications			
InstCreation-ForFanAdded	CIM_InstCreation	Conditional	See X-7.4.31.
InstDeletion-ForFanRemoved	CIM_InstDeletion	Conditional	See X-7.4.32 .
AlertForFanAdded	CIM_AlertIndication	Conditional	See X-7.4.34 .
AlertForFanRemoved	CIM_AlertIndication	Conditional	See X-7.4.35 .
AlertForFanFailed	CIM_AlertIndication	Optional	See X-7.4.36 .
AlertFor-FanReturnedToOK	CIM_AlertIndication	Optional	See X-7.4.37 .
AlertForFanDegraded	CIM_AlertIndication	Optional	See X-7.4.38 .

A.3 Example of a Description clause

Table 18 shows an example of the Description clause for an Example Fan profile.

3387 **Table 18 – Example of a Description clause**

X-6 Description

The Example Fan profile addresses the management domain of representing and managing fans in managed systems, including:

- the representation of the relationship between fans and the elements that are provided cooling by the fan
- the representation of sensors measuring the revolution speed of fans
- fan state management

Figure <Fig1> represents the DMTF collaboration structure diagram the Example Fan profile.

NOTE Here one or more DMTF collaboration diagram would be placed. For examples, see Figure 6 on page 53 .

Systems containing fans are modeled through the FanSystem adaptation, fans are modeled through the Fan adaptation, and their relationship is model through the SystemDevice association adaptation.

The element for that a fan provides cooling is modeled through the CooledElement adaptation, and the relationship between a fan and the managed element for that the fan provides cooling is modeled through the AssociatedCooling adaptation.

Fan speed sensoring is modeled through a conditional FanSpeedSensor feature; its implementation is only required for managed environments containing fans with speed sensors. As part of the FanSpeedSensor feature Fan sensors are modeled either through the DiscreteFanSpeedSensor adaptation (in case of sensors measuring discrete revolution speed values such as "high", "normal" or "low"), or through the AnalogFanSpeedSensor adaptation (in case of sensors measuring an analogous speed value in RPM), and the relationship between a fan and its sensors is modeled through the AssociatedSensor association adaptation.

The management of fan state is modeled through an optional FanStateManagement feature. The FanStateManagement feature requires the implementation of the RequestStateChange( ) method for the Fan adaptation.

3388 ## A.4 Example of an Implementation clause

3389 ### A.4.1 Example of the general layout of an Implementation clause

3390 Table 19 shows an example of the general layout of the Implementation clause; see 9.4.7 for require-
3391 ments on the specification of the Implementation clause.

3392 **Table 19 – Overview example of an Implementation clause**

X-7 Implementation

X-7.1 General

…

// general implementation requirements

…

X-7.2 Features

// see A.4.2 for example definitions of features

X-7.3 Events

// see A.4.3 for example definitons of events

X-7.4 Adaptations

// see A.4.4 for an example of the "Profile conventions for operations" subclause

// see A.4.5 for examples of subclauses defining adaptations of ordinary classes and associations

// see A.4.6 for examples of subclauses defining indication filter requirements

// see

A.4.2 Example of feature definitions

Table 20 shows examples of feature definitions within the Features subclause of the Implementation subclause; see 7.12 for requirements on the specification of features.

Table 20 – Example definitions of features

X-7.2.1 Feature: FanStateManagement

The implementation of the FanStateManagement feature is conditional.

Condition: The managed environment includes fans that are state manageable.

X-7.2.1.1 Feature description

The implementation of the FanStateManagement feature enables clients to request state changes on fans, such as activation or deactivation.

X-7.2.1.2 Feature discovery

The presence of the FanStateManagement feature for a particular Fan instance (see X-7.4.3) is indicated by the exposure of a FanCapabilities instance (see X-7.4.5) that is associated to the Fan instance through a FanElementCapabilities association instance (see X-7.4.6), and the value of the RequestedStatesSupported[ ] array property in the FanCapabilities instance is a non-empty list of values, each representing a supported requestable state for the fan.

X-7.2.2 Feature: FanElementNameEdit

[not detailed in this example]

…

X-7.2.3 Feature: FanSpeedSensor

The implementation of the FanSpeedSensor feature is conditional.

Condition: The managed environment includes fans with sensors.

X-7.2.3.1 Feature description

Fan speed sensoring is the capability of a fan to provide information about its revolution speed. Fan speed sensor information may be reported as discrete values such as "Normal", or as analogous speed such as "1200" rpm.

X-7.2.3.2 Feature discovery

The presence of the FanSpeedSensor feature for a particular Fan instance (see X-7.4.3) is indicated by the exposure of a FanSensor instance (see X-7.4.7) that is associated to the Fan instance through a SensorOfFan instance (see X-7.4.9), and the Sensors profile is supported for the FanSensor instance.

…

X-7.2.4 Feature: FanLifecycleIndications

The implementation of the FanLifecycleIndications feature is conditional.

Condition: The managed environment provides for the dynamic addition and removal of fans.

This feature defines the requirements for reporting fan lifecycle events (see X-7.3.1 and X-7.3.2) through instance indications.

The presence of the FanLifecycleIndications feature is indicated by a FilterCollection instance in the Interop namespace; see X-7.4.20 . Additionally, the presence of the implementation of individual indications that are defined as part of the FanLifecycleIndications feature may indicated by IndicationFilter instances in the Interop namespace; see X-7.4.21 .

X-7.2.5 Feature: FanProcessIndicationsForLifecycleEvents

The implementation of the FanProcessIndicationsForLifecycleEvents feature is optional.

The FanProcessIndicationsForLifecycleEvents feature groups the requirements for reporting fan lifecycle events (see X-7.3.1 and X-7.3.2) through process indications.

The presence of the FanProcessIndicationsForLifecycleEvents feature is indicated by a FilterCollection instance in the Interop namespace; see X-7.4.20 .

Additionally, the presence of the implementation of individual indications defined as part of the FanProcessIndicationsForLifecycleEvents feature may indicated by IndicationFilter instances in the Interop namespace; see X-7.4.21 .

A.4.3 Example of event definitions

Table 21 shows examples of event definitions within the Events subclause of the Implementation subclause; see 7.8 for requirements on the specification of events.

Table 21 – Example definitions of events

X-7.3.1 FanAdded event

The Fan Added event indicates the addition of a fan to the managed environment. After a fan was added to the managed environment the provisions of X-7.4.3 apply.

X-7.3.2 FanRemoved event

The FanRemoved event indicates the removal a fan from the managed environment. After a fan was removed from the managed environment the provisions of X-7.4.3 no longer apply and the fan shall not be represented as defined in X-7.4.3 .

X-7.3.3 FanFailed event

The FanFailed event indicates the failure of a fan. After a failure a fan should continued to be represented as defined in X-7.4.3; however, a failure may affect the implementations ability to obtain data about the fan that ultimately may lead to the implementation no longer being able to represent the fan by a CIM_Fan instance.

X-7.3.4 FanReturnedToOK event

The FanReturnedToOK event indicates the situation where a fan returns to its normal operational state after a failure; the provisions of X-7.4.3 apply.

X-7.3.5 FanDegraded event

The FanDegraded event indicates the situation where a fan is unable to provide the required cooling power; the provisions of X-7.4.3 apply.

A.4.4 Example of "Profile conventions for operations" subclause

Table 22 details an example of the "Profile conventions for operations" subclause within the Adaptations subclause of the Implementation clause; see 9.4.7.4.2 for requirements on the specification of implementation requirements for operations.

Table 22 – Example of "Profile conventions for operations" subclause

X-7.4.1 Profile conventions for operations

This profile defines operations in terms of DSP0223.

The implementation requirements on intrinsic operations (including on those listed in the following default lists) are specified in adaptation specific subclauses of 7.3 .

The default list of intrinsic operations for adaptations for ordinary classes is:

- GetInstance( )
- EnumerateInstances( )
- EnumerateInstanceNames( )

The default list of intrinsic operations for adaptations of association classes is:

- GetInstance( )
- EnumerateInstances( )
- EnumerateInstanceNames( )
- Associators( )
- AssociatorNames( )
- References( )
- ReferenceNames( )

…

A.4.5 Examples of subclauses defining adaptations

Table 23 details examples of subclauses within the Adaptation subclause of the Implementation clause that define adaptations of ordinary classes and associations; see 9.4.7.4 for requirements on the specification of class adaptations.

Table 23 – Examples of subclauses defining adaptations

X-7.4.3 Adaptation: Fan: CIM_Fan

This subclause defines the Fan adaptation for the representation of fans in systems.

X-7.4.3.1 Implementation requirements

The Fan adaptation adapts the CIM_Fan class.

The Fan adaptation is based on the CentralElement adaptation defined in DSP1033 (Profile Registration profile).

If the FanSpeedSensor feature (see X-7.2.3) is implemented, the Fan adaptation is based on the MonitoredElement adaptation of the Example Sensors profile.

Each fan in the managed environment shall be represented by a Fan instance.

Table X-10 lists the implementation requirements for the Fan adaptation.

Table X-10 – Adaptation: Fan: CIM_Fan

Element	Requirement	Description
OperationalStatus[ ]	Mandatory	See CIM schema definition
HealthState	Mandatory	See CIM schema definition
VariableSpeed	Mandatory	See CIM schema definition
DesiredSpeed	Conditional	Condition: The FanSpeedSensor feature is implemented; see X-7.2.3 . See CIM schema definition
ActiveCooling	Mandatory	Value shall be TRUE
EnabledState	Mandatory	See CIM schema definition
RequestedState	Conditional	Condition: The FanStateManagement feature is implemented; see X-7.2.1 . See CIM schema definition
ElementName	Conditional	Condition: The FanElementNameManagement feature is implemented; see X-7.2.2 . See CIM schema definition
RequestStateChange( )	Conditional	Condition: The FanStateManagement feature is implemented; see X-7.2.1 . See CIM Schema definition

All operations listed in the default list of operations for adaptations of ordinary classes (see X-7.4.1) shall be implemented as defined in DSP0223. Table X-11 lists additional operations that may be implemented as defined by DSP0223.

Table X-11 – Fan operations

Operation	Requirement	Messages
ModifyInstance( )	Optional; see 7.3.3.3	(1) DMTF.SMWG0001 (4) DMTF.SMWG0101 (4) DMTF.SMWG8001 For details see X-7.4.3.6 .

X-7.4.3.2 Property: EnabledState

The value of the EnabledState property shall convey the state of the represented fan. Admissible values are 2 (Enabled) and 3 (Disabled); all other values shall not be used. A value of 2 (Enabled) shall convey that the fan is activated and working, a value of 3 (Disable) shall convey that the fan is inactive.

X-7.4.3.3 Property: RequestedState

The value of the RequestedState property shall convey the most recently requested or desired state of the represented fan. Admissible values are 2 (Enabled) and 3 (Disabled); all other values shall not be used. A value of 2 (Enabled) shall convey that the fan is desired to be activated, a value of 3 (Disable) shall convey that the fan is desired to be inactive.

X-7.4.3.4 Method: RequestStateChange( )

The implementation of the RequestStateChange( ) method for the Fan adaptation is conditional.

Condition: The FanStateManagement feature is implemented; see X-7.2.1 .

If the RequestStateChange( ) method is implemented, the provisions in this subclause apply.

The behavior of the method shall depend on the value of the RequestedState parameter; this is referred to as the *requested state* in this subclause. The Fan instance on that the method is invoked is referred to as the *target instance* in this subclause. The fan in the managed environment that is represented by the target instance is referred to as the *target fan* in this subclause.

The method semantics shall be as follows:

- The value of the RequestedState property in the target instance shall reflect the last requested state.
- If the requested state is 2 (Enabled), the implementation shall request activation of the target fan.
- If the requested state is 3 (Disabled), the implementation shall request deactivation of the target fan.
- Any other requested state shall be rejected, issuing message DMTF.SMWG8009.
- Depending on the outcome of the operation requested by the implementation, the resulting state shall be reflected by the value of the EnabledState property.

X-7.4.3.5 Operation: ModifyInstance( )

The implementation of the ModifyInstance( ) operation for the Fan adaptation is optional.

If the ModifyInstance( ) operation is implemented, the provisions in this subclause apply.

The behavior of the method shall depend on the Fan instance that is passed in as the value of the ModifiedInstance parameter; this is referred to as the *input instance* in this subclause. The value of the EnabledState property in the input instance is referred to as the *requested state* in this subclause. The key properties in the input instance shall be used to identify the Fan instance for that the modification is requested; this instance is referred to as the *target instance* in this subclause. The fan in the managed environment that is represented by the target instance is referred to as the *target fan* in this subclause.

The method semantics with respect to the requested state shall be identical to those defined for the RequestStateChange( ) method; see X-7.4.3.4 .

This profile does not specify the implementation behavior regarding other properties of the input instance.

X-7.4.3.6 Standard messages

Table X-12 specifies the standard messages returned by the RequestStateChange( ) method.

Table X-12 – Fan standard messages

(return) MessageID	Message
(1) DMTF.SMWG0001	Not implemented
(4) DMTF.SMWG8001	Disable fan state change request failed because the temperature of one or more effected element is too high.

…

X-7.4.4 Adaptation: FanInSystem: CIM_SystemDevice

This subclause defines the FanInSystem adaptation of the CIM_SystemDevice association for the representation of the relationship of fans and their containing system.

Table X-15 lists the implementation requirements for the FanInSystem adaptation.

Table X-15 – Adaptation: FanInSystem: CIM_SystemDevice

Element	Requirement	Description

GroupComponent	Mandatory	**Key**: Value shall reference the CIM_System instance representing the system that contains the fan **Multiplicity**: 1
PartComponent	Mandatory	**Key**: Value shall reference the CIM_Fan instance representing a fan **Multiplicity**: *

All operations listed in the default list of operations for adaptations of association classes (see X-7.4.1) in shall be implemented as defined in DSP0223.

Each Fan instance required by X-7.4.3 shall be associated to the System instance representing the scoping system through a FanInSystem instance.

X-7.4.5 Adaptation: FanCapabilities: CIM_EnabledLogicalElementCapabilities

This subclause defines the FanCapabilities adaptation of the CIM_EnabledLogicalElementCapabilities class for the representation of the capabilities of fans in managed systems.

The implementation of the FanCapabilities adaptation is conditional.

Condition: One or more of the following conditions:

- The FanStateManagement feature is implemented; for feature definition see X-7.2.1 .
- The FanElementNameEdit feature is implemented; for feature definition see X-7.2.2 .

X-7.4.5.2 Implementation requirements

Table X-12 list the requirement for this class adaptation.

Table X-12 – Adaptation: FanCapabilities: CIM_EnabledLogicalElementCapabilities

Element	Requirement	Description
RequestedStatesSupported[ ]	Conditional	Condition: The FanStateManagement feature is implemented; see X-7.2.1 . See CIM schema definition.
ElementNameEditSupported	Conditional	Condition: The ElementNameEdit feature is implemented; see X-7.2.2 . Value shall be TRUE.
MaxElementNameLen	Conditional	Condition: The ElementNameEditSupported property is implemented. See CIM schema definition.

All operations listed in the default list of operations for adaptations of ordinary classes (see **X-**7.4.1) shall be implemented as defined in DSP0223.

X-7.4.5.2 Instance requirements

If FanCapabilities is implemented, the capabilities of a fan represented by a Fan instance (see X-7.4.3) may be represented by a FanCapabilities instance.

X-7.4.6 Adaptation: CapabilitiesOfFan: CIM_ElementCapabilities

This subclause defines the CapabilitiesOfFan adaptation of the CIM_ElementCapabilities association for the representation of the relationship between a fan and its capabilities.

The implementation of the CapabilitiesOfFan adaptation is conditional.

Condition: The FanCapabilities adaptation is implemented; see X-7.4.5 .

Example-Table 13 lists the requirements for this association adaptation.

Example-Table 13 – Adaptation: CapabilitiesOfFan: CIM_ElementCapabilities

Element	Requirement	Description
ManagedElement	Mandatory	**Key**: Value shall reference the Fan instance representing a fan **Multiplicity**: 1..*
Capabilities	Mandatory	**Key**: Value shall reference the CIM_EnabledLogicalElement instance representing the fans capabilities **Multiplicity**: 0..1

All operations listed in the default list of operations for adaptations of association classes (see **X-**7.4.1) shall be implemented as defined in DSP0223.

Each FanCapabilities instance (see X-7.4.5) shall be associated to its Fan instance (see X-7.4.3) through a CapabilitiesOfFan instance.

X-7.4.7 Adaptation: FanSensor: CIM_Sensor

This subclause defines the FanSensor adaptation of the CIM_Sensor class for the representation of fan speed sensors. The FanSensor adaptation shall be based on the Sensor adaptation defined in DSPxxxx (Example Sensors profile).

The implementation of the FanSensor adaptation is conditional.

Condition: All of the following:

- The FanSpeedSensor feature is implemented
- Fan speed sensors within the managed environment support reporting discrete speed only.

Table X-14 lists the requirements for this class adaptation.

Table X-14 – Adaptation: FanSensor: CIM_Sensor

Element	Requirement	Description
SensorType	Mandatory	Value shall be 5 (Tachometer)

All operations listed in the default list of operations for adaptations of ordinary classes (see **X-**7.4.1) shall be implemented as defined in DSP0223.

Each fan speed sensor within the managed environment that supports reporting discrete speed only shall be represented by a FanSensor instance.

X-7.4.8 Adaptation: FanNumericSensor: CIM_NumericSensor

This subclause defines the FanNumericSensor adaptation of the CIM_NumericSensor class for the representation of fan speed sensors.

The implementation of the FanNumericSensor adaptation is conditional.

Condition: All of the following:

- The FanSpeedSensor feature is implemented
- Fan speed sensors within the managed environment support reporting analogous speed.

Table X-15 lists the requirements for this class adaptation.

Table X-15 – Adaptation: FanNumericSensor: CIM_NumericSensor

Elements	Requirement	Notes

SensorType	Mandatory	Value shall be 5 (Tachometer)
BaseUnits	Mandatory	Value shall be 19 (RPM)
RateUnits	Mandatory	Value shall be 0 (None)

All operations listed in the default list of operations for adaptations of ordinary classes (see **X-**7.4.1) shall be implemented as defined in DSP0223.

Each fan speed sensor within the managed environment that supports reporting analogous speed shall be represented by a FanNumericSensor instance.

X-7.4.9 Adaptation: SensorOfFan: CIM_AssociatedSensor

This subclause defines the SensorOfFan adaptation of the CIM_AssociatedSensor association for the representation of the relationship between a fans and their sensors.

The implementation of the SensorOfFan adaptation is conditional.

Condition: The FanSpeedSensor feature is implemented; for feature definition see **X-**7.2.3 .

Table X-16 lists the requirements for this association adaptation.

Table X-16 – Adaptation: SensorOfFan: CIM_AssociatedSensor

Element	Requirement	Description
Antecedent	Mandatory	**Key**: Value shall reference the FanSensor (see X-7.4.7) or the FanNumericSensor (see X-7.4.8) instance representing the sensor attached to the fan. **Multiplicity**: 1
Dependent	Mandatory	**Key**: Value shall reference the Fan instance representing a fan **Multiplicity**: *

All operations listed in the default list of operations for adaptations of association classes (see X-7.4.1) shall be implemented as defined in DSP0223.

Each Sensor instance representing a fan sensor as required by X-7.4.7 or X-7.4.8 shall be associated to the Fan instance representing the respective fan through a SensorOfFan instance.

…

A.4.6 Examples of subclauses defining indication filter requirements

Table 24 details examples of subclauses within the Adaptation subclause of the Implementation clause that define adaptations of classes modeling indication filters and filter collections.

Table 24 – Examples of subclauses defining indication filter requirements

…

X-7.4.20 Adaptation: FilterCollection: CIM_FilterCollection

This subclause defines the FilterCollection adaptation of the CIM_FilterCollection class for the representation of filter collections.

The implementation of the FilterCollection adaptation is conditional.

Condition: Any indication adaptation defined in this profile is implemented; see X-7.4.31, X-7.4.32, X-7.4.34, X-7.4.35, X-7.4.36, X-7.4.37 and X-7.4.38 .

X-7.4.20.1 Implementation requirements

Table X-20 lists the requirements for this class adaptation; these requirements are in addition to those specified in DMTF:DSP1054:1.0 (*Indications* profile).

Table X-20 – Adaptation: FilterCollection: CIM_FilterCollection

Element	Requirement	Description
CollectionName	Mandatory	See X-7.4.20 .

X-7.4.20.2 Instance requirements

Table X-21 lists the requirements for FilterCollection instances in the Interop namespace. If and only if the condition stated in a cell in the Condition column in Table 21 is met, an implementation shall expose a FilterCollection instance with the value of the CollectionName property as defined in the corresponding cell in the "Value of CollectionName property" column.

Table X-21 – CIM_FilterCollection for fan lifecycle indications

Condition	Value of CollectionName property
The FanLifecycleIndications feature is implemented; see X-7.2.4 .	"DMTF:Fan:FanLifecycleIndications"
The FanProcessIndicationsForLifecycleEvents feature is implemented; see X-7.2.5 .	"DMTF:Fan:FanProcessIndicationsForLifecycleEvents"

…

X-7.4.21 Adaptation: IndicationFilter: CIM_IndicationFilter

This subclause defines the IndicationFilter adaptation of the CIM_IndicationFilter class for the representation of indication filters.

The implementation of the IndicationFilter adaptation is conditional.

Condition: One or more of adaptations of the CIM_AlertIndication indication as defined in X-7.4.36, X-7.4.37 and X-7.4.38 are implemented.

The CIM_IndicationFilter class should be implemented if any indication adaptation is implemented; see X-7.4.31, X-7.4.32, X-7.4.34, X-7.4.35, X-7.4.36, X-7.4.37 and X-7.4.38 .

X-7.4.21.1 Implementation requirements

Table X-22 lists the requirements for this class adaptation; these requirements are in addition to those specified in DSP1054 (*Indications* profile).

Table X-22 – Adaptation: IndicationFilter: CIM_IndicationFilter

Element	Requirement	Description
Name	Mandatory	See Table X-23 .
Query	Mandatory	See Table X-24 .

X-7.4.21.2 Implementation requirements

The provisions in this subclause aim at establishing client discovery of the implementation of specific indication adaptations. The implementation of a particular indication adaptation is indicated by the presense of a corresponding IndicationFilter instance in the Interop namespace. The implementation of a particular group of related indication adaptations is indicated by the presence of a corresponding FilterCollection instance in the Interop namespace. If the group is small, provisions in this subclause only require the FilterCollection instance.

Table X-23 defines the relationship between indication adaptations and IndicationFilter instances in the

Interop namespace.

Example-Table X-23 – Relationship between IndicationFilter instances and indication adaptation implementations

CIM_IndicationFilter.Name	Related indication adaptation	
"DMTF:Fan:FanAdded"	InstCreationForFanAdded; see X-7.4.31 .	
"DMTF:Fan:FanRemoved"	InstDeletionForFanRemoved; see X-7.4.32 .	
"DMTF:Fan:DMTF:PLAT456"	AlertForFanAdded; see X-7.4.34 .	
"DMTF:Fan:DMTF:PLAT457"	AlertForFanRemoved; see X-7.4.35 .	
"DMTF:Fan:DMTF:PLAT458"	AlertForFanFailed; see X-7.4.36 .	
"DMTF:Fan:DMTF:PLAT459"	AlertForFanReturnedToOK; see X-7.4.37 .	
"DMTF:Fan:DMTF:PLAT460"	AlertForFanDegraded; see X-7.4.38 .	

Table X-24 defines the relationship between IndicationFilter instances and FilterCollection instances.

Table X-24 – Relationship between IndicationFilter instances and FilterCollection instances

CIM_IndicationFilter.Name	CIM_FilterCollection.CollectionName
"DMTF:Fan:FanAdded"	"DMTF:Fan:FanLifecycleIndications"
"DMTF:Fan:FanRemoved"	"DMTF:Fan:FanLifecycleIndications"
"DMTF:Fan:DMTF:PLAT456"	"DMTF:Fan:FanProcessIndicationsForLifecycleEvents"
"DMTF:Fan:DMTF:PLAT457"	"DMTF:Fan:FanProcessIndicationsForLifecycleEvents"
"DMTF:Fan:DMTF:PLAT458"	"DMTF:Conditional/Optional"
"DMTF:Fan:DMTF:PLAT459"	"DMTF:Conditional/Optional"
"DMTF:Fan:DMTF:PLAT460"	"DMTF:Conditional/Optional"

If the indication adaptation listed in a cell of the "Related indication adaptation" column in Table X-23 is is implemented, the following provisions apply for a corresponding IndicationFilter instance:

- If the value in the corresponding cell in Table X-24 of the CollectionName column identifies the "DMTF:Conditional/Optional" filter collection, the implementation *shall* expose an IndicationFilter instance in the Interop namespace, with the value of the Name property as defined in Table X-23 and the value of the Query property as defined in Table X-25.
- If the value in the corresponding cell in Table X-24 of the CollectionName column does not identify the "DMTF:Conditional/Optional" filter collection, the implementation *should* expose an IndicationFilter instance in the Interop namespace, with the value of the Name property as defined in Table X-23 and the value of the Query property as defined in Table X-25.

Table X-25 lists the requirements for the value of the Query property in IndicationFilter instances.

Table X-25 – Value of the Query property in IndicationFilter instances

CIM_IndicationFilter.Name	CIM_IndicationFilter.Query
"DMTF:Fan:FanAdded"	"Select * From CIM_InstCreation Where SourceInstance ISA CIM_Fan"
"DMTF:Fan:FanRemoved"	"Select * From CIM_InstDeletion Where SourceInstance ISA CIM_Fan"
"DMTF:Fan:DMTF:PLAT456"	"Select * From CIM_AlertIndication Where OwningEntity = DMTF And MessageID=PLAT456"
"DMTF:Fan:DMTF:PLAT457"	"Select * From CIM_AlertIndication Where OwningEntity = DMTF And Mes-

	sageID=PLAT457"
"DMTF:Fan:DMTF:PLAT458"	"Select * From CIM_AlertIndication Where OwningEntity = DMTF And MessageID=PLAT458"
"DMTF:Fan:DMTF:PLAT459"	"Select * From CIM_AlertIndication Where OwningEntity = DMTF And MessageID=PLAT459"
"DMTF:Fan:DMTF:PLAT460"	"Select * From CIM_AlertIndication Where OwningEntity = DMTF And MessageID=PLAT460"

NOTE The values of the Query property defined in Table X-25 reflect summarized descriptions for the generation and assembly of indications; these requirements are detailed in other subclauses of this profile. For example a value of "Select * From CIM_InstCreation Where SourceInstance ISA CIM_Fan" describes that a CIM_InstCreation indication is to be generated when a CIM_Fan instance is created. Implicit assumption here is that a CIM_Fan instance is "created" when a fan is added to the managed system. This profile renders this requirement more precisely by defining the FanAdded event to mark the addition of a fan to a managed system, and in this case requiring the implementation to a) expose an instance of the Fan adaptation (see X-7.4.3) and to b) generate an InstCreationForFanAdded indication adaptation instance (see X-7.4.31).

X-7.4.22 Adaptation: MemberOfFilterCollection: CIM_MemberOfCollection

This subclause defines the MemberOfFilterCollection adaptation of the CIM_MemberOfCollection association for the representation of the relationship between a filter collections and the collected filters.

The implementation of the MemberOfFilterCollection adaptation is conditional.

Condition: The IndicationFilter adaptation is implemented; see X-7.4.21 .

Table X-26 lists the requirements for this association adaptation; these requirements are in addition to those specified in DSP1054 (Indications profile).

Table X-26 – Adaptation: MemberOfFilterCollection: CIM_MemberOfCollection

Element	Requirement	Description
Collection	Mandatory	**Key**: Value shall reference the FilterCollection instance representing the filter collection **Multiplicity**: 1
Member	Mandatory	**Key**: Value shall reference the IndicationFilter instance representing a collected indication filter **Multiplicity**: *

Each IndicationFilter instance required by X-7.4.21 shall be associated with a FilterCollection instance required by X-7.4.20 through a MemberOfFilterCollection instance.

If an IndicationFilter instance is instantiated in the Interop namespace (see X-7.4.21), and the value of the CIM_IndicationFilter.Name property matches a value listed in the CIM_IndicationFilter.Name column in Table X-24, that IndicationFilter instance shall be associated through a MemberOfFilterCollection instance to the FilterCollection instance (see X-7.4.20) for which the value of the CIM_FilterCollection.CollectionName property matches that specified in the corresponding cell in the CIM_FilterCollection.CollectionName column in Table X-24.

A.4.7 Examples of subclauses defining indication adaptations

Table 25 details examples of subclauses within the Adaptation subclause of the Implementation clause that define adaptations of indications.

3421 **Table 25 – Examples of subclauses defining indication adaptations**

X-7.4.30 General requirements: CIM_Indication

This subclause details general requirements for the implementation of adaptations of the CIM_Indication indication.

X-7.4.30.1 General

Table X-41 lists general requirements for adaptations of the CIM_AlertIndication indication; these requirements are in addition to those specified in DSP1054 (*Indications* profile).

Table X-41 – Common requirements for adaptations of the CIM_AlertIndication indication

Element	Requirement	Description
IndicationIdentifier	Mandatory	Value shall contain a unique identification of the indication instance; for the format see CIM schema definition.
IndicationTime	Mandatory	Value shall contain the time of the reported event; for the format see CIM schema definition.
PerceivedSeverity	Mandatory	Value shall be identical to that of the PERCEIVED_SEVERITY xml property in the message definition of the message identified by the value of the MessageID property.
IndicationFilterName	Mandatory	See X-7.4.30.2 .

X-7.4.30.2 Property: CIM_Indication.IndicationFilterName

The value of the IndicationFilterName property in any generated indication instance shall be set as follows:

If a related CIM_IndicationFilter instance is instantiated in the Interop namespace for the indication (see X-7.4.21), then the value of the IndicationFilterName property in the CIM_Indication instance shall be identical to that of the Name property in the related CIM_IndicationFilter instance; otherwise, the value of the IndicationFilterName property in the CIM_Indication instance shall be identical to the value of the CollectionName property in the related CIM_FilterCollection instance instantiated in the Interop namespace (see X-7.4.20).

X-7.4.31 Adaptation: InstCreationForFanAdded: CIM_InstCreation

The implementation of the InstCreationForFanAdded adaptation is conditional.

Condition: The FanLifecycleIndications feature is implemented; for feature definition see X-7.2.4 .

Example-Table 42 lists the requirements for this indication adaptation; these requirements are in addition to those specified in X-7.4.30 .

Table X-42 – Adaptation: InstCreationForFanAdded: CIM_InstCreation

Element	Requirement	Description
SourceInstance	Mandatory	Value shall contain a copy of the Fan instance representing the added fan; see X-7.4.3 .
SourceInstanceModelPath	Mandatory	Value shall refer to the Fan instance representing the added fan; see X-7.4.3. For the format of the reference see CIM schema definition.
SourceInstanceHost	Optional	Value may contain the host name or IP address of the system hosting the added fan.

A InstCreationForFanAdded instance shall be generated for each FanAdded event; see X-7.3.1 .

X-7.4.32 Adaptation: InstDeletionForFanRemoved: CIM_InstDeletion

The implementation of the InstDeletionForFanRemoved indication adaptation is conditional.

Condition: The FanLifecycleIndications feature is implemented; for feature definition see X-7.2.4 .

Table X-43 lists the requirements for this indication adaptation; these requirements are in addition to those specified in X-7.4.30 .

Table X-43 – Adaptation: InstDeletionForFanRemoved: CIM_InstDeletion

Element	Requirement	Description
SourceInstance	Optional	Value shall contain a copy of the Fan instance that represented the removed fan; see X-7.4.3 . NOTE: The Fan instance no longer exists.
SourceInstanceModelPath	Mandatory	Value shall refer to the Fan instance that represented the removed fan; see X-7.4.3. For the format of the reference see CIM schema definition. NOTE: The Fan instance no longer exists.
SourceInstanceHost	Optional	Value may contain the host name or IP address of the system hosting the added fan.

A InstDeletionForFanRemoved instance shall be generated for each FanRemoved event; see X-7.3.2 .

X-7.4.33 Adaptation: AlertIndication: CIM_AlertIndication

The AlertIndication adaptation details general requirements for the implementation of adaptations of the CIM_AlertIndication indication as specified in X-7.4.34, X-7.4.35, X-7.4.36, X-7.4.37 or X-7.4.38 . The requirements defined in this subclause are in addition to those specified in DMTF DSP1054:1.0 (*Indications* profile).

Table X-44 lists general requirements for adaptations of the CIM_AlertIndication indication; these requirements are in addition to those specified in X-7.4.30 .

Table X-44 – Common requirements for adaptations of the CIM_AlertIndication indication

Element	Requirement	Description
AlertingManagedElement	Mandatory	Value shall refer to the CIM_Fan instance representing the fan in context of that the reported event occurred.
AlertingManagedElement-Format	Mandatory	Value shall be 3 (Wbem URI)
AlertType	Mandatory	Value shall be identical to that of the ALERT_TYPE xml property in the message definition of the message identified by the value of the MessageID property.
MessageID	Mandatory	Value specified in specific adaptation; see X-7.4.34, X-7.4.35, X-7.4.36, X-7.4.37 or X-7.4.38 .
Message	Optional	Value specified in specific adaptation; see X-7.4.34, X-7.4.35, X-7.4.36, X-7.4.37 or X-7.4.38 .
MessageArguments	Mandatory	Value specified in specific adaptation; see X-7.4.34, X-7.4.35, X-7.4.36, X-7.4.37 or X-7.4.38 .

X-7.4.34 Adaptation: AlertForFanAdded: AlertIndication

The implementation of the AlertForFanAdded indication adaptation is conditional.

Condition: The FanProcessIndicationsForLifecycleEvents feature is implemented; for feature definition see **X-**7.2.4 .

X-7.4.34.1 Implementation requirements

The AlertForFanAdded indication adaptation specializes the AlertIndication adaptation; see X-7.4.33 .

Table X-45 lists the requirements for this indication adaptation; these requirements are in addition to those specified in X-7.4.33 .

Table X-45 – Adaptation: AlertForFanAdded: CIM_AlertIndication

Element	Requirement	Description
MessageID	Mandatory	Value shall match "PLAT456".
Message	Optional	If implemented, value shall contain the complete text of message "PLAT456", with actual argument values.
MessageArguments[0]	Mandatory	Value shall be identical to the value of the ElementName property in the Fan instance representing the added fan; see X-7.4.3 .
MessageArguments[1]	Mandatory	Value shall be in WBEM URI format and refer to the CIM_ComputerSystem instance representing the scoping computer system.

X-7.4.34.1 Indication generation requirements

If subscriptions exist, an AlertForFanAdded instance shall be generated for each FanAdded event; see X-7.3.1 .

X-7.4.35 Adaptation: AlertForFanRemoved: AlertIndication

The implementation of the AlertForFanRemoved indication adaptation is conditional.

Condition: The FanProcessIndicationsForLifecycleEvents feature is implemented; for feature definition see **X-**7.2.5 .

The AlertForFanRemoved indication adaptation specializes the AlertIndication adaptation; see X-7.4.33 .

Table X-46 lists the requirements for this indication adaptation; these requirements are in addition to those specified in X-7.4.33 .

Table X-46 – Adaptation: AlertForFanRemoved: CIM_AlertIndication

Element	Requirement	Description
MessageID	Mandatory	Value shall match "PLAT457".
Message	Optional	If implemented, value shall contain the complete text of message "PLAT457", with actual argument values.
MessageArguments[0]	Mandatory	Value shall be identical to the value of the ElementName property in the Fan instance that represented the removed fan; see X-7.4.3 . NOTE: The Fan instance no longer exists.
MessageArguments[1]	Mandatory	Value shall be in WBEM URI format and refer to the CIM_ComputerSystem instance representing the scoping computer system.

An AlertForFanRemoved instance shall be generated for each FanRemoved event; see X-7.3.2 .

X-7.4.36 Adaptation: AlertForFanFailed: CIM_AlertIndication

The implementation of the AlertForFanFailed indication adaptation is optional.

Table X-47 lists the requirements for this indication adaptation; these requirements are in addition to

those specified in X-7.4.33 .

Table X-47 – Adaptation: AlertForFanFailed: CIM_AlertIndication

Element	Requirement	Description
MessageID	Mandatory	Value shall match "PLAT458".
Message	Optional	If implemented, value shall contain the complete text of message "PLAT458", with actual argument values.
MessageArguments[0]	Mandatory	Value shall be identical to the value of the ElementName property in the Fan instance representing the failed fan; see X-7.4.3 .
MessageArguments[1]	Mandatory	Value shall be in WBEM URI format and refer to the CIM_ComputerSystem instance representing the scoping computer system.

An AlertForFanFailed instance shall be generated for each FanFailed event; see X-7.3.3 .

X-7.4.37 Adaptation: AlertForFanReturnedToOK: CIM_AlertIndication

The implementation of the AlertForFanReturnedToOK indication adaptation is optional.

Table X-48 lists the requirements for this indication adaptation; these requirements are in addition to those specified in X-7.4.33 .

Table X-48 – Adaptation: AlertForFanReturnedToOK: CIM_AlertIndication

Element	Requirement	Description
MessageID	Mandatory	Value shall match "PLAT459".
Message	Optional	If implemented, value shall contain the complete text of message "PLAT459", with actual argument values.
MessageArguments[0]	Mandatory	Value shall be identical to the value of the ElementName property in the CIM_Fan instance representing the failed fan.
MessageArguments[1]	Mandatory	Value shall be in WBEM URI format and refer to the CIM_ComputerSystem instance representing the scoping computer system.

An AlertForFanReturnedToOK instance shall be generated for each FanReturnedToOK event; see X-7.3.4 .

X-7.4.38 Adaptation: AlertForFanDegraded: CIM_AlertIndication

The implementation of the AlertForFanDegraded indication adaptation is optional.

Table X-49 lists the requirements for this indication adaptation; these requirements are in addition to those specified in X-7.4.33 .

Table X-49 – Adaptation: AlertForFanDegraded: CIM_AlertIndication

Element	Requirement	Description
MessageID	Mandatory	Value shall match "PLAT460".
Message	Optional	If implemented, value shall contain the complete text of message "PLAT460", with actual argument values.
MessageArguments[0]	Mandatory	Value shall be identical to the value of the ElementName property in the CIM_Fan instance representing the failed fan.
MessageArguments[1]	Mandatory	Value shall be in WBEM URI format and refer to the CIM_ComputerSystem instance representing the scoping

		computer system.
An AlertForFanDegraded instance shall be generated for each FanDegraded event; see X-7.3.5 .		

A.5 Example of Use-cases clause

Table 26 provides an example of the Use-cases profile specification clause.

Table 26 – Example of Use-cases clause

X-8 Use-cases

…

X-8.3 Determine fan state

This use-case describes the use of the GetInstance( ) operation as adapted by this profile (see X-8.2.2) inspecting the state of a fan.

X-8.3.1 Preconditions

The client knows the instance path of the Fan instance representing the fan.

X-8.3.2 Flow of activities

1) The client obtains the Fan instance, invoking the GetInstance( ) operation with parameter values set as follows:
   - The value of the InstancePath parameter is set to the input instance path that refers to the Fan instance.
   - Optionally, the value of the IncludedProperties[ ] array property may be set to one element whose value is "EnabledState"; this would reduce the returned instance to include only the value of the EnabledState property.

The implementation executes the operation as requested by the client.

If the GetInstance( ) operation returns, the use-case continues with step 1)2) .

If the GetInstance( ) operation causes an exception, the use-case continues with step 1)4) .

2) The client inspects the return code
   - A return code of 0 indicates successful execution of the intrinsic operation; the use-case continues with step 3) .
   - A return code of 1 (Not Supported) indicates that the implementation does not support the method; this terminates the use-case, the postconditions in X-8.3.3.2 apply.
   - A return code of 2 (Unknown or Unspecified Error) indicates an error situation that is not covered by the profile specification; this terminates the use-case, the postconditions in 9.3.3.2 apply.
3) The client inspects the value of the EnabledState property of the returned CIM_Fan instance:
   - A value of 0 (Unknown) indicates that the state of the fan is unknown; this may be a temporary condition.
   - A value of 2 (Enabled) indicates that the fan is active.
   - A value of 3 (Disabled) indicates that the fan is inactive.
   - A value of 4 (Shutting Down) indicates that the fan is in the process of deactivating.
   - A value of 10 (Starting) indicates that the fan is in the process of activating.

- Other values are not adapted by this profile.

This completes the use-case; the postconditions in X-8.3.3.1 apply.

4) The GetInstance( ) intrinsic operation caused an exception. The client inspects the CIM_Error instances returned as part of the exception.

X-8.3.3 Postconditions

This subclause lists possible situations after the use-case execution.

X-8.3.3.1 Success

The fan state as reflected by the value of the EnabledState property is known to the client.

X-8.3.3.2 Failure

9) The fan state could not be determined; reasons were reflected through either through the value of the return code or through CIM_Error instances delivered as part of an exception.

…

X-8.7 Enabling a fan through the RequestStateChange( ) method

This use-case describes the use of the RequestStateChange( ) method as adapted by this profile (see X-8.1.1) for enabling a fan.

X-8.7.1 Preconditions

- The client knows the instance path of the CIM_Fan instance representing the fan.
- Fan state changes are supported for that instance (for detection see X-9.4) and the fan is currently disabled (for inspection see X-8.3).

X-8.7.2 Flow of activities

1) The client requests activation of the fan, invoking the RequestStateChange( ) method on the input instance representing the fan, with parameter values set as follows:
   - The value of the RequestedState property is 2 (Enabled)
   - The value of the TimeoutPeriod property is not provided (NULL)

The implementation executes the method as requested by the client.

If the RequestStateChange() method returns, the use-case continues with step 2) .

If the RequestStateChange() method causes an exception, the use-case continues with step 3) .

2) The client inspects the return code:
   - A return code of 0 indicates successful execution of the method. This completes the use-case; the post-conditions in X-8.7.4.1 apply.
   - A return code of 1 (Not Supported) indicates that the implementation does not support the method; this terminates the use-case, the postconditions in X-8.7.4.2 apply.
   - A return code of 2 (Unknown or Unspecified Error) indicates an error situation that is not covered by the profile specification; this terminates the use-case, the postconditions in X-8.7.4.3 apply.
   - A return code of 4 (Failed) indicates that the implementation was unable to enable the fan; this terminates the use-case, the postconditions in X-8.7.4.2 apply.
   - A return code of 5 (Invalid Parameter) indicates that one or more of the input parameters were invalid; this terminates the use-case, the postconditions in X-8.7.4.2 apply.
   - A return code of 6 (In Use) indicates that the fan is in use by another management activity; this terminates the use-case, the postconditions in X-8.7.4.3 apply.

- A return code of 4096 (Method Parameter Checked – Job Stared) indicates that an asynchronous task was started that performs and controls the fan state change operation that is represented by a CIM_ConcreteJob instance referenced by the value of the Job output parameter; the use-case continues with step 4).
- A return code of 4097 (Invalid State Transition) indicates that the fan is in a state that (presently) does not allow a transition to the requested state; this terminates the use-case, the postconditions in X-8.7.4.2 apply.

3) The RequestStateChange() method caused an exception. The client inspects the CIM_Error instances returned as part of the exception. This terminates the use-case, the postconditions in X-8.7.4.2 apply.

4) The client obtains the CIM_ConcreteJob instance, invoking the GetInstance( ) operation with parameter values set as follows:

- The value of the InstancePath parameter is set to value of the Job output parameter returned from step 1) .

The implementation executes the intrinsic operation as requested by the client.

If the GetInstance( ) intrinsic operation returns, the use-case continues with step 5) .

If the GetInstance( ) intrinsic operation causes an exception, the client inspects the CIM_Error instances returned as part of the exception. This terminates the use-case, the postconditions in X-8.7.4.2 apply.

5) The client inspects the value of the JobState property:

- A value of 7 (Completed) indicates successful execution of the use-case. This completes the use-case; the post-conditions in X-8.7.4.1 apply.
- A value matching { 2 | 3 | 4 | 5 | 11 | 12 } (New | Starting | Running | Suspended | Service | Query pending) indicates that the asynchronous task has not yet finished; after waiting a certain delay, the client continues with repeating step 4).
- Any other value matching indicates an error situation or a situation not anticipated in this profile; this terminates the use-case, the postconditions in X-8.7.4.2 apply.

X-8.7.4 Postconditions

This subclause list possible situations after the use-case execution.

X-8.7.4.1 Success

- The fan is enabled
- If inspected for example by performing use-case X-8.3, the value of the EnabledState property in the instance of the CIM_Fan class representing the fan has the value 1 (Enabled).

NOTE The client should regularly validate (for example through the application of use-case X-8.3) that the fan remains enabled, as conditions in the managed environment (failures, activities by other operators, etc.) could cause fan state changes. Alternatively the client could monitor CIM_InstModification indications indicating state changes in the CIM_Fan instance representing the fan.

X-8.7.4.2 Failure with unchanged state

The fan remains disabled.

X-8.7.4.2 Failure with undefined state

The state of the fan is undetermined.

3425

ANNEX B Regular expression syntax

This annex defines the regular expression syntax used in profile specifications to specify the format of values, especially those representing identifiers. The regular expression grammar below uses Augmented BNF (ABNF) as defined in RFC5234 with the following exceptions:

- Ranges of alphabetic characters or numeric values may be specified using two periods ("..") placed between the beginning and ending values of the range (in addition to the use of the minus sign (-) as defined in ABNF).
- The rules defined in this syntax are assembled into a complete query by assuming white space characters between them, except where noted otherwise. (ABNF requires explicit specification of white space.)

NOTE 1 ABNF is NOT case-sensitive.

NOTE 2 The implicit concatenation rules as specified by ABNF apply, that is, in any ABNF rule in this annex, whitespace outside of literals is not part of the definition and to be ignored. However, this does NOT apply to the regular expression that are defined in profiles conforming to this ABNF. In particular, except where noted, white space is significant within these conforming regular expressions.

Profile regular expressions are a subset of the regular expressions defined in UNIX Regular Expressions.

The following elements are defined:

Special characters

```
SpecialChar = "." / "\" / "[" / "]" / "^" / "$" / "*" / "+" / "?" /
"/" / "|"
```

where

"."	matches any single character
"\"	escapes the next character so that it isn't a `SpecialChar`
"["	starts a `CharacterChoice`
"]"	ends a `CharacterChoice`
"^"	indicates a `LeftAnchor`
"$"	indicates a `RightAnchor`
"*"	indicates that the preceding item is matched zero or more times.
"+"	indicates that the preceding item will be matched one or more times.
"?"	indicates that the preceding item is optional and will be matched at most once.
"|"	separates choices

Ordinary characters

```
OrdinaryChar = UnicodeChar, except SpecialChar
```

where

`UnicodeChar` refers to any Unicode character, as defined in RFC3629.

Escaped special characters

```
EscapedChar = "\" SpecialChar
```

Simple character

```
SimpleChar = OrdinaryChar / EscapedChar
```

Character sequence

```
CharacterSequence = SimpleChar | CharacterSequence
```

A `CharacterSequence` is a sequence of `SimpleChar`s, such as for example,

`"ABC"` matching `"ABC"`, or

`"D.F"` matching `"DAF"`, `"DBF"`, `"DCF"`, and so forth.

Character choice

```
CharacterChoice = "[" CharacterSequence "]" [ "^" ]
```

A `CharacterChoice` defines a set of possible characters. It is indicated by square brackets (`"["` and `"]"`) enclosing the set of characters.

– If a caret (`"^"`) is *not* suffixed after the closing bracket, any character from the set matches. For example, "r[au]t" matches "rat" or "rut".

– If a caret (`"^"`) is suffixed after the closing bracket, any character *not* in the set matches. For example, "r[au]^t" matches any 3 character sequence with the middle character not being `"a"` or `"u"`, such as for example `"ret"` or `"r.t"`.

Single character

```
SingleChar = "." / SimpleChar / CharacterChoice
```

For example,

`"D.F"` matching `"DAF"`, `"DBF"`, `"DCF"`, and so forth, or

`"GH[IJ]"` matching `"GHI"` or `"GHJ"`.

Multipliers

```
Multiplier = "*" / "+" / "?" / "{" UnsignedInt ["," [UnsignedInt]] "}"
```

where

`"*"` indicates that the preceding item is matched zero or more times

`"?"` indicates that the preceding item is matched zero or one time (optional item)

`"+"` indicates that the preceding item is matched one or more times.

`UnsignedInt` is an unsigned integer number.

Multiplied character

```
MultipliedChar = SingleChar [ Multiplier ]
```

A `MultipliedChar` is a `SingleChar` with a `Multiplier` applying, such as for example

`"C*"` matching `""`, `"C"`, `"CC"`, `"CCC"`, and so forth, or

`"[EF]{1,2}"` matching `"E"`, `"F"`, `"EE"`, `"EF"`, `"FE"` or `"FF"`

Character expression

```
CharacterExpression = MultipliedChar | MultipliedChar CharacterExpression
```

A `CharacterExpression` is a descriptor for a sequence of one or more characters, such as for example

`"X"` matching `"X"` only,

3503 `"ABC"` matching `"ABC"` only,

3504 `"ABC*"` matching `"AB"`, `"ABC"`, `"ABCC"`, `"ABCCC"`, and so forth,

3505 `"A[BC]D"` matching `"ABD"` or `"ACD"`, or

3506 `"1[.]{2,3}n"` matching `"1..n"` or `"1...n"`.

3507 **Grouping**

3508 `Grouping = "(" SimpleChoice ")" [ Multiplier ]`

3509 A `Grouping` is a `SimpleChoice` that optionally can be multiplied, such as for example

3510 `"(ABC)"` matching `"ABC"`,

3511 `"(XYZ)+"` matching `"XYZ"`, `"XYZXYZ"`, `"XYZXYZXYZ"`, and so forth.

3512 **Choice**

3513 `Choice = Grouping | Grouping Choice`

3514 A `Choice` is a choice from one or more `Grouping`s, such as for example,

3515 `"(DEF)?|GHI"` matching `""`, `"DEF"`, or `"GHI"`

3516 **Left anchor**

3517 `LeftAnchor = "^"`

3518 A `LeftAnchor` forces a match at the beginning of a string.

3519 **Right anchor**

3520 `RightAnchor = "$"`

3521 A `RightAnchor` forces a match at the end of a string.

3522 **AnchoredExpression**

3523 `AnchoredExpression = [ RightAnchor ] Choice [ LeftAnchor ]`

3524 An `AnchoredExpression` is a `Choice` that is optionally anchored to the left end, to the right
3525 end or to both ends of a string.

3526 **AnchoredChoice**

3527 `AnchoredChoice = AnchoredExpression "|" AnchoredExpression Anchored-`
3528 `Choice`

3529 An `AnchoredChoice` is a choice from one or more `AnchoredExpression`s.

3530 **RegularExpressionInProfile**

3531 `RegularExpressionInProfile = AnchoredChoice`

3532 A regular expression within a profile is an `AnchoredChoice`.

ANNEX C (informative) Bibliography

This clause lists references that are helpful for the application of this guide.

UML Specifications,
http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML

UML Intro: Practical UML, A Hands-In Introduction for Developers,
http://bdn.borland.com/article/0,1410,31863,00.html

DMTF DSP1000, *Management Profile Specification Template 1.1*
http://www.dmtf.org/standards/published_documents/DSP1000_1.1.pdf

ANNEX D (informative) Change history

Version	Date	Description
1.0.0	2006-06-14	Initial final release
1.0.1	2009-08-05	DMTF Standard Release. Changes: Updated copyright statement Updated and corrected references listed in 2 Added provisions for specifying a scoping algorithm in 6.1 Simplified and corrected profile conventions for operations in 6.4.2 Added Annex F, Experimental Content Added Annex G, Change Log Added Bibliography Minor text corrections throughout the document.
1.1.0k	2009-11-03	Work in progress release. Changes: New concepts: Adaptations, features and events Rules for the definition of indications Rules for defining the relationship to the managed environment Condensed structure of profile specifications Many clarifications and corrections