Document number: DSP1001

Date: 2011-06-30

Version: 1.1.0

Management Profile Specification Usage Guide

Document type: Specification

Document status: DMTF Standard

Document language: us-EN

CONTENTS

Figures

Tables

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. For information about the DMTF, see http://www.dmtf.org.

Acknowledgments

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

- Jim Davis, WBEM Solutions
- George Ericson, EMC
- Steve Hand, Symantec
- Jon Hass, Dell
- Michael Johanssen, IBM
- Andreas Maier, IBM
- Aaron Merkin, Dell
- Karl Schopmeyer, DMTF Fellow
- Paul von Behren, Sun Microsystems

Introduction

The information in this guide should be sufficient for profile authors to incorporate all the semantic and formal elements required for the specification of a management profile. The information in this guide should be sufficient for profile implementers to ascertain the implementation requirements imposed by this guide, by the set of implemented profiles, by the CIM schema and by other appropriate specifications.

Document conventions

Typographical conventions

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

- Document titles are marked in *italics*.[1]
- Important terms that are used for the first time are marked in *italics*.
- Terms include a link to the term definition in the "Terms and definitions" clause, enabling easy navigation to the term definition.
- ABNF rules are in `monospaced font`.

ABNF usage conventions

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-insensitive US-ASCII characters.
- The following ABNF rules are frequently applied in this guide:

```
CR = %x0D

CRLF = CR LF

HTAB = %x09

LF = %x0A

LWSP = *( WSP / CRLF WSP)

SP = %x20

WS = 1*WSP

WSP = SP / HTAB
```

Deprecated material

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

[1] Note that referencing a profile by its name does not constitute a document title; for details, see 7.6.2.

backwards compatibility. Although CIM clients may use deprecated elements, they are directed to use the favored elements instead.

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

The following typographical convention indicates deprecated material:

DEPRECATED

Deprecated material appears here.

DEPRECATED

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

Experimental material

Experimental material has yet to receive sufficient review to satisfy the adoption requirements set forth by the DMTF. Experimental material is included in this document as an aid to implementers who are interested in likely future developments. Experimental material may change as implementation experience is gained. It is likely that experimental material will be included in an upcoming revision of the document. Until that time, experimental material is purely informational.

The following typographical convention indicates experimental material:

EXPERIMENTAL

Experimental material appears here.

EXPERIMENTAL

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

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 server and a WBEM client. In addition, a profile may define a management interface between a WBEM server 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 (including indication delivery operations) 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 target 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 DSP0215, *Server Management Managed Element Addressing Specification 1.0*, http://www.dmtf.org/standards/published_documents/DSP0215_1.0.pdf

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

DMTF DSP0228, *Message Registry XML Schema 1.1*, http://www.dmtf.org/standards/published_documents/DSP0228_1.1.xsd

DMTF DSP1033, *Profile Registration Profile 1.0*, http://www.dmtf.org/standards/published_documents/DSP1033_1.0.pdf

DMTF DSP1053, *Base Metrics Profile 1.1*,
http://www.dmtf.org/standards/published_documents/DSP1053_1.1.pdf

DMTF DSP1054, *Indications Profile 1.1*,
http://www.dmtf.org/standards/published_documents/DSP1054_1.1.pdf

DMTF DSP4004, *DMTF Release Process 2.3*,
http://www.dmtf.org/standards/published_documents/DSP4004_2.3.pdf

DMTF DSP8016, *WBEM Operations Message Registry 1.0*,
6http://schemas.dmtf.org/wbem/messageregistry/1/dsp8016_1.0.xml

DMTF DSP8020, *Message Registry XML Schema Specification 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, Part 2:2004, *Rules for the structure and drafting of International Standards*,
http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype

Object Management Group, OMG UML Superstructure, *OMG Unified Modeling Language (OMG UML) Superstructure 2.1.2*

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

3 Terms and definitions

In this guide, some terms and verbal phrases have a specific meaning beyond the normal English meaning. Those terms and verbal phrases are defined in this clause.

The verbal phrases "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, Part 2, Annex H. The verbal phrases in parenthesis are alternatives for the preceding verbal phrase, for use in exceptional cases when the preceding verbal phrase cannot be used for linguistic reasons. Note that ISO/IEC Directives, Part 2, 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, Part 2, Clause 5.

The terms "normative" and "informative" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, 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.

3.1
abstract
a possible implementation type of class adaptations
For details, see 7.13.5.

3.2

abstract class adaptation

a class adaptation with an implementation type of "abstract".

The requirements of abstract class adaptations apply only in the context of other class adaptations that use them as base adaptations.

For details, see 7.13.5.

3.3

abstract profile

a special kind of profile specifying common elements and behavior as a base for derived profiles

For a complete definition, see 7.9.2.11.

3.4

adaptation

short form for class adaptation

3.5

adaptation instance

an instance of an adapted class that complies with all requirements of the class adaptation

For details see 5.3.

3.6

adapted class

a class that is the subject of a class adaptation

For details, see 7.13.

3.7

autonomous profile

a profile that addresses an autonomous and self-contained management domain

For details, see 7.8.2.

3.8

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 the context of a profile implementation of a later minor version, without requiring modifications of the client

For a complete definition, see 7.17.

3.9

base adaptation

a class adaptation that is used as the base for another class adaptation

For details, see 7.13.2.1.

3.10

base profile

a profile that is used as the base for another profile

For details, see 7.9.1 and 7.9.2.

3.11

central class adaptation

a specifically designated class adaptation in a profile

The central class adaptation is the focal point of the profile. For a complete definition, see 7.9.3.2.

3.12

class

if used without qualification this term refers to a CIM class that may also be an association class or an indication class. To refer to a CIM class that is not an association class or an indication class, 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 7.13.

3.14

client

a WBEM client that exploits applicable portions of a profile

See also the term "implementation".

3.15

component profile

a profile that addresses a subset of a management domain

For details, see 7.8.3.

3.16

concrete profile

any profile that is not an abstract profile

For a complete definition, see 7.10.2.

3.17

concrete class adaptation

any class adaptation that is not an abstract class adaptation

For details, see 7.13.5.

3.18

condition

a specification mechanism in profiles that determines whether conditional or conditional exclusive profile elements shall be implemented

For a complete definition, see 7.4.

3.19

conditional

a requirement level indicating that the subject profile requires the implementation of the designated profile element only under certain conditions, and otherwise leaves the decision to implement the designated profile element to the implementation

See 7.3 for usage considerations, and 9.2 for implementation considerations.

3.20

conditional exclusive

a requirement level indicating that the subject profile requires the implementation of the designated profile element only under certain conditions, and otherwise prohibits the implementation of the designated profile element

See 7.3 for usage considerations, and 9.2 for implementation considerations.

3.21

conditional profile

a used profile that is referenced by a profile reference with the conditional requirement level

3.22

conditional exclusive profile

a used profile that is referenced by a profile reference with the conditional exclusive requirement level

3.23

deprecated

keyword indicating that a profile element or profile defined behavior is outdated and has been replaced by newer constructs

For details, see 7.17.

3.24

derived profile

a profile that is based on a referenced profile

For a complete definition, see 7.9.2.

3.25

discovery mechanism

a CIM based mechanism yielding a Boolean result that enables clients to discover whether optional, conditional or conditional exclusive profile elements are implemented or available

For a complete definition, see 7.5.

3.26

error reporting requirement

a requirement stated as part of a method requirement or operation requirement to report an error situation

For details, see 7.13.3.2.4 and 7.13.3.3.6.

3.27

event

an observable occurrence of a phenomenon of interest

For details, see 6.7.

3.28

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.

3.29

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 7.15.

3.30

implementation

a WBEM server that implements applicable portions of one or more profiles

For example, in server-side infrastructures using CIM providers, implementation refers to the WBEM server and the set of providers that implement applicable portions of the set of profiles, that is, the implementation adaptation set.

For details, see clause 9.

3.31

implementation adaptation

an implementation-required adaptation that merges the requirements of its base adaptations and of other sources such as the schema definition of the adapted class, the operations specification or registry elements

For a complete definition, see 9.2.2.

3.32

implementation adaptation set

the set of implementation adaptations required to be implemented as part of an implementation

For a complete definition, see 9.2.1.

3.33

implementation-required

a phrase indicating that the implementation of a profile or profile element is required within an implementation, including the case where an optional profile or profile element was selected to be implemented

For a complete definition, see 9.2.1.

3.34

implementation type

a type assigned to an adaptation that details how the adaptation is to be implemented

For a complete definition, see 7.13.2.5.

3.35

incompatibility

a change that breaks backward compatibility

3.36

indication

the notification about an event that occurred

3.37

indication adaptation

an adaptation of an indication class

3.38

indication-generation requirement

a requirement that states one or more events (see 6.7), each of which individually requires the generation of a particular indication

For details, see 7.13.4.2.

3.39

input value requirement

a requirement stated as part of a property requirement, or of a parameter requirement within a method requirement, that requires that the implementation accepts a specific input value

For details, see 7.13.2.11.

3.40

instance requirement

a requirement that defines how (and in some cases also under which conditions) managed objects are to be represented by adaptation instances

For details, see 7.13.3.4.

3.41

listener

a WBEM listener that implements applicable portions of the Indications profile (see DSP1054)

3.42

management domain

area of work or field of activity with common management requirements, common terminology, and related management functionality

For details, see 6.2.

3.43

managed environment

a concrete occurrence of the management domain. A managed environment is composed of managed objects

For details, see 6.4.

3.44

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.

3.45

managed object type

a conceptual generalization or type of managed object

For details, see 6.3.

3.46

management profile

definition of a management interface between a WBEM server and a WBEM client or a WBEM listener

For a complete definition, see clause 1.

3.47

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.

3.48

mandatory

a requirement level indicating that the subject profile unconditionally requires the implementation of the designated profile element

See 7.3 for usage considerations, and 9.2 for implementation considerations.

3.49

mandatory profile

a used profile that is referenced by a profile reference with the mandatory requirement level

3.50

match

keyword indicating that a property or parameter value is within the values specified by a pattern

For details see 10.2.4.

3.51

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 7.13.3.2.

3.52

message registry

a published registry of messages formatted as defined in DSP0228

3.53

metric requirement

a requirement stated as part of a class adaptation that defines requirements and constraints on a metric defined in a metric registry

For details, see 7.13.3.5.

3.54
metric registry
a published registry of metric definitions, and optionally statistics definitions, formatted as defined in DSP8020

3.55
named profile element
a profile element that is assigned a name with profile name scope
For details, see 7.2.2.

3.56
operation requirement
a requirement stated as part of a class adaptation that defines requirements and constraints on an operation defined in an operations specification
For details, see 7.13.3.3.

3.57
operations specification
a specification that specifies operations, their semantics and the model and behavior associated to them
Examples are DSP0223 and DSP0200.

3.58
optional
a requirement level indicating that the subject profile leaves the decision to implement the designated profile element to the implementation
See 7.3 for usage considerations, and 9.2 for implementation considerations.

3.59
optional profile
a used profile that is referenced by a profile reference with the optional requirement level

3.60
ordinary class
a class that is not an association class or an indication class
For a complete definition, see DSP0004.

3.61
organization
in this guide, refers to a consortium, standards group, company, or business entity creating a management profile

3.62
pattern
specification of the permissible values for a property or parameter
See also the term "match", and for details see 10.2.4.

3.63
profile
synonym for management profile
See 3.46, and for a complete definition, see clause 1.

3.64
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.

3.65
profile derivation
profile derivation establishes a referenced profile as the base profile of the referencing profile
For details, see 7.9.1 and 7.9.2.

3.66
profile element
formal elements that this guide establishes to be specified by profiles
For a complete definition, see 7.2.

3.67
profile implementation
a subset of an implementation that realizes the requirements of a particular profile in a particular profile implementation context

3.68
profile implementation context
a context in which a profile or an adaptation is implemented
For a complete definition, see 9.2.3.

3.69
profile specification
synonym for management profile specification
See 3.47, and for a complete definition see clause 1.

3.70
profile reference
a named profile element that references another profile
For details, see 7.9.1.

3.71
profile usage
a use of the referenced profile established by a referencing profile
For details, see 7.9.1.

3.72
prohibited
a requirement level indicating that the subject profile prohibits the implementation of the designated profile element
See 7.3 for usage considerations, and 9.2 for implementation considerations.

3.73
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 7.13.2.8.

3.74
referenced profile
a profile that is referenced by another profile, establishing either profile derivation or a profile usage
For a complete definition, see 7.9

3.75
referencing profile
a profile that references another profile, establishing either profile derivation or a profile usage
For a complete definition, see 7.9.

3.76

registry reference

a named profile element referencing a message registry or a metric registry

For details, see 7.12.

3.77

related profile

deprecated synonym for referenced profile

3.78

requirement level

designator that indicates the requirement for implementing profile elements or used profiles

3.79

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)

3.80

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 association classes and indication classes), properties (including references), methods, and parameters

3.81

scoping class adaptation

a specifically designated class adaptation in a profile that is the algorithmic focal point for identifying profile conformance when using the scoping class methodology.

For a complete definition, see 7.9.3.3.

3.82

scoped profile

a profile that receives a scope provided by a scoping profile. Synonymous with component profile

For details, see 7.9.3.

3.83

scoping path

an association traversal path between the central class adaptation and the scoping class adaptation.

For details, see 7.9.3.4.

3.84

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 7.9.3.

3.85

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 7.13.2.1.

3.86

state description

a named profile element that describes of the state of an instance of (a subset of) the model defined by a profile at a particular point in time

For a complete definition, see 7.16.2.

3.87

subject profile

a profile created or verified in conformance to this guide

3.88

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 10.4.7.4.

3.89

use case

a named profile element that defines an interaction of an external client and an implementation in the execution of steps required to be performed in the realization of functionality defined in a profile

For details, see 7.16.

3.90

used profile

a referenced profile that is used by the referencing profile

3.91

WBEM client

a CIM client (see DSP0004) that supports a WBEM protocol

A WBEM client originates WBEM server operations. This definition does not imply any particular implementation architecture or scope, such as a client library component or an entire management application. For details, see DSP0223.

3.92

WBEM listener

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

A WBEM listener processes WBEM listener operations. This definition does not imply any particular implementation architecture or scope, such as a client library component or an entire management application. For details, see DSP0223.

3.93

WBEM protocol

a communications protocol between WBEM client, WBEM server and WBEM listener

A WBEM protocol defines how the WBEM operations work, on top of an underlying protocol layer (for example, HTTP, SOAP, or TCP). For details, see DSP0223.

3.94

WBEM server

a CIM server (see DSP0004) that supports a WBEM protocol

A WBEM server processes WBEM server operations, and originates WBEM listener operations. This definition does not imply any particular implementation architecture, such as a separation into generic and adaptation-specific (provider) components. For details, see DSP0223.

4 Symbols and abbreviated terms

Most of these symbols and abbreviated terms are also applicable 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

ACID

atomicity, consistency, isolation, and durability

4.2

CSD

DMTF collaboration structure diagram

For details, see 8.3.4.

4.3

PUG

Profile Usage Guide (the usage guide for specifying profiles specified in this document, DSP1001)

4.4

UFcT

User Friendly class Tag, as defined in DSP0215

4.5

UFiT

User Friendly instance Tag, as defined in DSP0215

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 and in clause 8.

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 10 .

5.2 Implementation conformance

5.2.1 Interface implementation conformance

A profile implementation 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 referred to as black box testing.

Examples of requirements defined only in terms of the model are as follows:

- Value constraints that restrict a property value to a set of possible values, such as 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

NOTE However, is should be noted that if such a test is performed by creating the instance in a first step, and obtaining the instance in a second step, it is absolutely possible that the instance was already modified or deleted again after the first step, but before the second step is performed. For that reason a more realistic test is checking the dependency between the instance and the managed object that it represents. See 5.2.2 for white box testing, and see also 6.6.2 for the existence of instances.

Examples of requirements that are not defined only in terms of the model are as follows:

- 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 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 the activation of a managed object in the case where a change of the EnabledState property to 2 (Enabled) in the CIM instance representing the managed object is requested

5.2.2 Full implementation conformance

Full implementation conformance extends interface implementation conformance by also considering profile defined requirements that establish the relationship of the profile defined model and the managed environment.

Full implementation conformance can be validated only by crosschecking the situation in the managed environment with the situation as viewed through the profile defined interface. Consequently, the validation of full implementation conformance 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, for example, class adaptations in one profile being based on managed environments 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 7.9.1, a subject profile refers to referenced profiles by specifying only the major and minor version number, implying the latest published update versions of the referenced profiles. Consequently it is possible that various implementations of a comprehensive set of profiles (such as an identified version of a particular subject profile, and all its referenced profiles), that are created at different points in time, use different update versions of the referenced profiles.

For that reason, conformance of a *profile implementation* to a profile is defined only with regard to a specific update version of that profile.

For example, if a particular profile P1 references version 1.0 of P2, and if P1 was written when version 1.0.1 of a referenced profile P2 was published, at that time P1 would effectively reference version 1.0.1 of P2 and an implementation implementing P1 and P2 would have to implement version 1.0.1 of P2. When at a later point in time version 1.0.2 of P2 is published, from that time on P1 would effectively reference version 1.0.2 of P2, and an implementation implementing P1 and P2 would then have to implement version 1.0.2 of P2. Thus the first implementation conforms to version 1.0.1 of P2, and the second implementation conforms to version 1.0.2 of P2. The backward compatibility rules defined in 7.17 strive for only permitting changes that do not invalidate the second implementation to version 1.0.1 of P2; however — as detailed in 7.17 — it is possible that version 1.0.2 introduces incompatible changes as part of error corrections.

5.2.5 Listener implementation conformance

A WBEM listener is conformant to DSP1054 if it implements all requirements targeting WBEM listeners. Note that profiles implementing DSP1054 reference a particular version, and conformance is required with respect to that version.

Further, a conformant WBEM listener shall implement the indication delivery related listener operations defined in the operations specification. Note that this guide does not require that the same operations specification is selected for the communication between the WBEM server and the WBEM listener, and that between the WBEM client and the WBEM server.

5.2.6 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 implemented profiles and by other specifications referenced by them.

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 the context of an implementation is a complex process that must consider all involved sources of requirements, such as base adaptations, the CIM schema definition of the adapted class, and operations specifications; see clause 9 for a detailed description of that process.

5.4 DMTF conformance requirements

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

- Management profiles owned by DMTF shall conform to this guide. The normative requirements for profiles are detailed in clause 7 and in clause 8.
- Management profile specifications owned by DMTF shall conform to this guide. The normative requirements for profile specifications are detailed in clause 10. In addition, the standard DMTF specification format (see DSP1000) applies to DMTF-owned management profile specifications.

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

6 Concepts

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

6.1 Overview

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

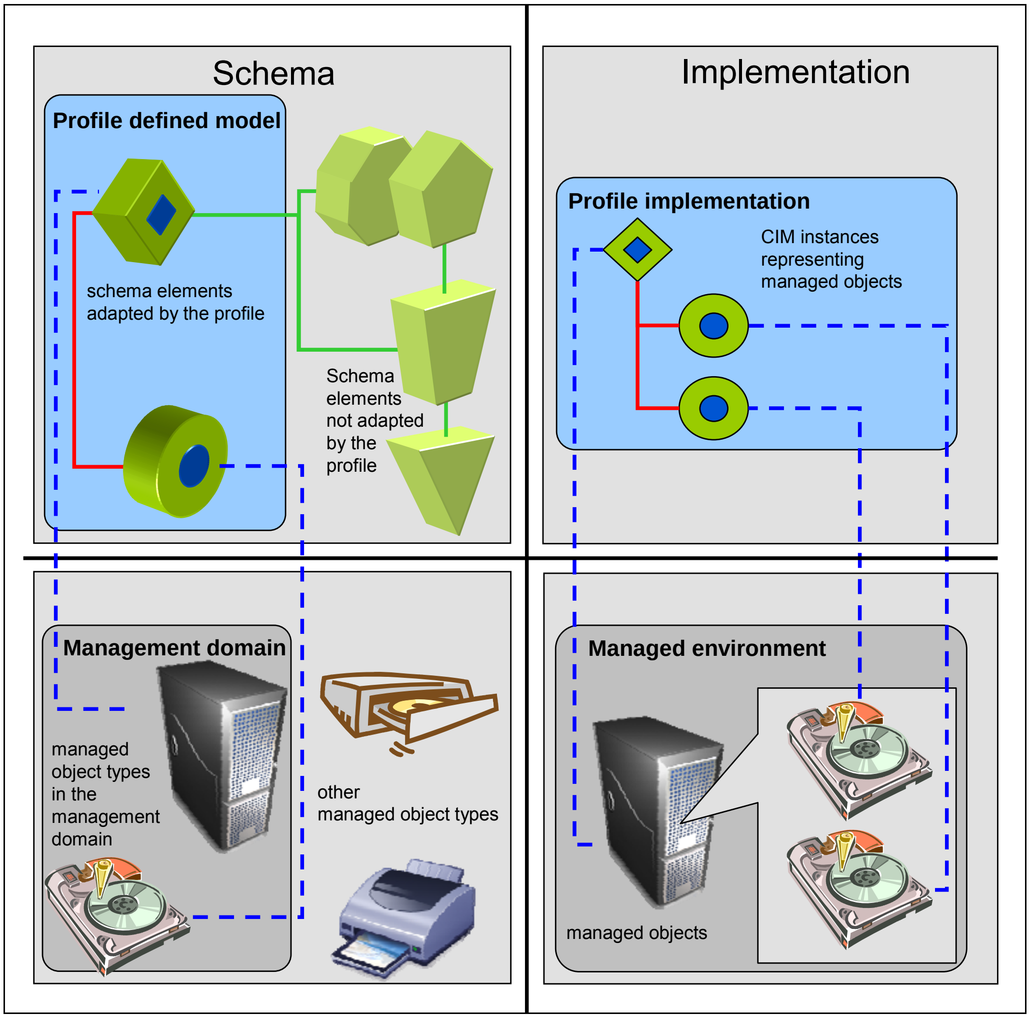

Schema
Profile defined model
schema elements adapted by the profile
Schema elements not adapted by the profile
Implementation
Profile implementation
CIM instances representing managed objects
Management domain
managed object types in the management domain
other managed object types
Managed environment
managed objects

Figure 1 – Profile and management domain

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

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

6.2 Management domain

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

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

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

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

6.3 Managed object type

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

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

6.4 Managed environment and managed objects

A *managed environment* is a concrete occurrence of a management domain and is composed of *managed objects*. For example, a managed environment within the file system management domain is a concrete Linux ext3 file system that resides on some storage media and is composed of objects such as the file system itself, its files, directories, links, access lists, or quotas. For a particular type of managed environment (for example, Linux ext3 file systems) specific management instrumentation (such as a set of commands, or an API) may exist that allow the inspection and manipulation of managed objects in respective managed environments. For example, instances of the Linux ext3 file system in a desktop installation may be inspected and manipulated through means of the Linux ext3 file system device 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 7.13.

6.6 Relationships between profile definition and management domain

6.6.1 Profile defined mappings

A profile defines the following mappings:

- the mapping between managed object types composing a management domain and class adaptations modeling (aspects of) these managed object types.

  This kind of mapping is established in profiles by means of defining the management domain addressed by the profile, particularly the managed object types in that management domain, and by further stating for each adaptation which (aspect of a) managed object type is modeled by that adaptation; for details, see 7.11 and 7.13.2.2.

- the mapping between managed objects composing a managed environment and adaptation instances representing aspects of these managed objects.

  This kind of mapping is established in profiles by means of instance requirements stated as part of the definition of adaptations; for details, see 7.13.3.4.

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 adaptation instances.

6.6.2 Existence and lifecycle of adaptation instances

In a managed environment the managed objects or relationships between them can 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 6.6.3, but in general the cause for such changes is outside the scope of the profile implementation.

Recall that adaptation instances are instances of CIM classes that conform to the requirements of a particular adaptation; see 3.5.

The *existence* of adaptation instances is a logical concept: A particular adaptation instance is defined to exist in a namespace of a particular WBEM server exactly as long as the managed object that is represented by that adaptation instance exists in the managed environment.

It is emphasized that the existence of adaptation instances is a *logical concept*; particularly, the existence of an adaptation instance does not imply that the WBEM server in context of that the instance exists is active or that the managed environment containing the managed object representing the adaptation instance is accessible by the implementation within the WBEM server. Consequently, existing instances are not required to be visible to the clients all time.

NOTE One reason for defining the existence of adaptation instances as a logical concept independent from the activity state of the related WBEM server is avoiding the re-creation of adaptation instances when the WBEM server restarts that — among other consequences — would require the generation of respective lifecycle indications.

The *creation* of an adaptation instance is defined to occur when the represented managed object is added to the managed environment. This can occur if either a pre-existing managed object is added to the managed environment, or if a managed object is created within the managed environment. The former is typical for tangible managed objects such as disk drives or fans, while the latter is typical for intangible managed objects such as files, log entries or virtual systems. The creation of an adaptation instance is also the event that triggers the generation of a respective lifecycle indication; see 6.7.

The *deletion* of an adaptation instance is defined to occur when the represented managed object is removed from the managed environment. This occurs as a managed object such as a hardware component is removed from the managed environment, but also if a managed object such as a database record is deleted and thus no longer exists as part of the managed environment. The deletion of an adaptation instance is also the event the triggers the generation of a respective lifecycle indication; see 6.7.

These interrelationships are detailed in Figure 2.

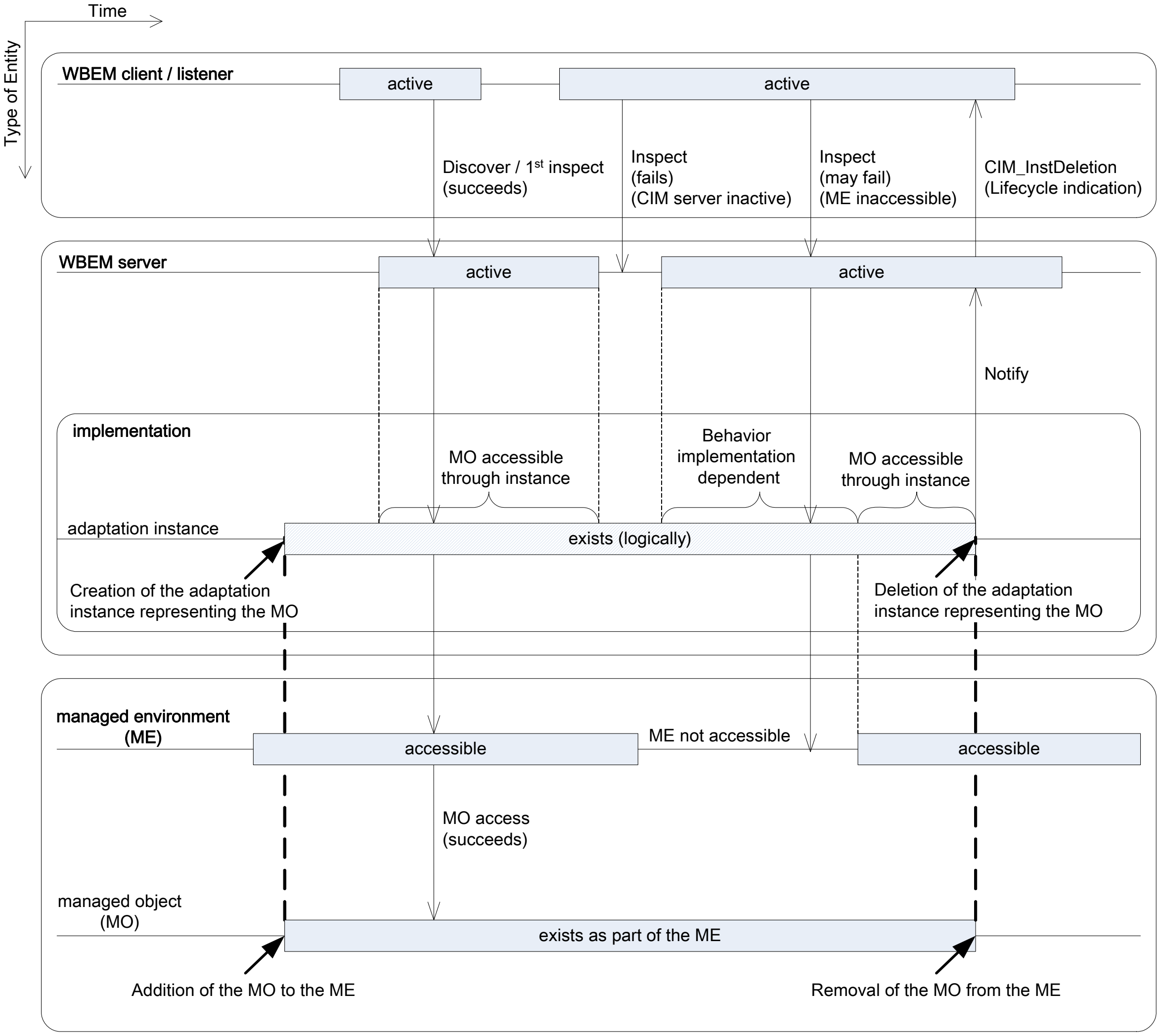


Figure 2 – Existence of adaptation instances

Figure 2 further details that the existence of an adaptation instance does not require that the WBEM server in context of that the instance exists is active. This implies that an existing adaptation instance is not all times accessible by clients. Various other reasons may also impede client access to adaptation instances, such as for example the implementation not being able to access the managed object in the managed environment.

All the information exposed by an adaptation instance originates from the represented managed object. While a managed object is not accessible by the implementation, the representing adaptation instance(s) should not expose imprecise, outdated or otherwise unsynchronized information about the current state of the managed object. In case of doubt an implementation should raise an error or otherwise indicate that the represented managed object is not accessible, or that certain property values are not available; for example, the special value Null can be used to indicate the absence of a value.

As a consequence, the only cause for a change in an adaptation instance is a respective change in the represented managed object. 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; for details, see 6.6.3.

NOTE There is much flexibility in defining managed object types. For example, it is possible for a profile to define managed object types such that configuration data is separated from functional data. That way an implementation could be realized such that configuration data is kept separately in a database and would be accessible while the database is accessible, whereas functional data would only be accessible if the functional part of a managed object is accessible; however, if a client requests a complete adaptation instance, the previously mentioned restrictions on exposing information apply also in this case with respect to the functional part.

Adaptation 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 for example repeatedly invoke the GetInstance( ) operation of the adaptation 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 6.7.

6.6.3 Model effected control of managed objects in a managed environment

CIM initiated modifications on the model are only actable if the represented managed environment admits such modifications. Profiles may define CIM-based control of managed objects in a managed environment by assigning management domain specific semantics to methods or operations defined by the model; for details, see 7.13.3.2 or 7.13.3.3. If such a method or operation is invoked, the implementation issues requests to the affected managed object in the managed environment in order to perform the profile 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.

Adaptation instances represent aspects of 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 or operations. 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 adaptation instance. This situation drives the need for profiles to define the means that indicate completion for model effected changes.

6.7 Events and indications

An event is an observable occurrence of a phenomenon of interest. Profiles specify events as part of indications. For details, see DSP1054.

Indications model notifications about events. Notifications about events that are related to CIM instances representing particular managed objects are modeled as *lifecycle indications*; notifications about other kinds of events are modeled through *alert indications*; for details, see DSP1054.

7 Profile definitions

7.1 General

Clause 7 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 8 defines general conventions and guidelines that apply for all kinds of profiles. Clause 10 defines the requirements for profile specification documents, focusing on formal text document aspects.

7.2 Profile elements

7.2.1 General

Profile elements are the (kinds of) formal elements that this guide establishes to be specified by profiles.

This guide defines following profile elements for the use in profiles:

- adaptations (see 7.13)
- features (see 7.15)
- profile references (see 7.9.1)
- registry references (see 7.12)
- property requirements (see 7.13.2.8)
- method requirements (see 7.13.3.2)
- operation requirements (see 7.13.3.3)
- input value requirements (see 7.13.2.11)
- error reporting requirements (see 7.13.3.3.6)
- state descriptions (see 7.16.2)
- use cases (see 7.16)

In many cases the requirements defined in a profile for a profile element are based on, refer to, extend or further constrain an entity that is defined outside of the profile. For example, an adaptation defined in a profile adapts a class defined in a schema for a particular purpose; or a registry reference refers to a registry of certain things such as messages or metrics, which are applied or used other definitions within the profile.

7.2.2 Named profile elements

The following profile elements are defined as named profile elements: adaptations, features, profile references, registry references, state descriptions and use cases.

A named profile element shall be assigned a name that uniquely identifies the named profile element within the scope of the profile defining the named profile element. Uniqueness is only required separately for each kind of named profile element; consequently, it is possible that within one profile for example a feature has the same name as an adaptation.

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.

NOTE This notation is occasionally termed camel-case notation (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 7.17 and 7.19 apply.

For example, StateManagement might name a feature that defines a model for the management of the state of managed objects. If version 1.0 had introduced that feature, subsequent minor versions would be required to retain the StateManagement feature under that name, and with identical or compatibly extended semantics. Subsequent minor versions could deprecate the feature, but only a new major version would be allowed to remove the feature.

Examples of adaptation names are Fan for an adaptation of the CIM_Fan class, or FanOfSystem 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 references defined by an Example Disk profile referencing an Example Sensors profile for the two purposes: The modeling of disk speed sensors and disk temperature sensors.

7.3 Usage of requirement levels

7.3.1 General

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

Occasionally individual requirement levels may be defined for specific purposes, such as the presentation, initialization or modification of adaptation instances.

The following requirement levels are defined:

- Mandatory, as defined in 3.48
- Optional, as defined in 3.58
- Conditional, as defined in 3.19
- Conditional exclusive, as defined in 3.20
- Prohibited, as defined in 3.72

It is emphasized that dependencies on other profile elements defined in the same or in other profiles, as well as dependencies on referenced definitions for example from referenced schemas or registries, may impose additional implementation requirements. The determination of implementation requirements and the effects of requirement levels with respect to the implementation requirements of profile elements are described in clause 9.

NOTE Requirement levels are formally defined only for the designation of profile elements (see 7.2). However, profiles may state other provisions such as instance requirements or indication-generation requirements using normative language (primarily verbal phrases such as "shall", "may", "should", etc.).

7.3.2 Usage of the "mandatory" requirement level

A subject profile should designate a profile element as mandatory if it unconditionally requires the implementation of the designated profile element. Clients can rely on mandatory profile elements being implemented once they have determined that the subject profile is implemented.

7.3.3 Usage of the "optional" requirement level

A subject profile should designate a profile element as optional if it leaves the decision to implement the profile element to the implementation. In other words, the implementation of an optional profile element is considered auxiliary or complementary from the perspective of the subject profile.

A CIM based discovery mechanism (see 7.5) should be defined that enables clients — after having determined that the subject profile is implemented — to determine whether the optional profile element is implemented. A CIM based discovery mechanism (see 7.5) shall be defined if other profile elements are defined as conditional or conditional exclusive on the optional profile element.

A profile that intends to define multiple optional profile elements that are useful to clients only as a group should define an optional feature (see 7.15) and define the elements as conditional on the implementation of that optional feature.

7.3.4 Usage of the "conditional" requirement level

A subject profile should designate a profile element as conditional if it requires the implementation of the designated profile element only under certain conditions, and otherwise leaves the decision to implement the designated profile element to the implementation.

For any profile element designated as conditional, the condition shall be defined using one of the mechanisms defined in 7.4.

A CIM based discovery mechanism (see 7.5) shall be defined that enables clients — after having determined that the subject profile is implemented — to determine whether the conditional profile element is available. The discovery mechanism may be defined indirectly, such that the discovery mechanism for one conditional profile element by means of conditional dependencies is delegated to that of another profile element; particularly, this is the case with feature implementation conditions (see 7.4.3) and feature discovery (see 7.15.6).

7.3.5 Usage of the "conditional exclusive" requirement level

A subject profile should designate a profile element as conditional exclusive if it requires the implementation of the designated profile element only under certain conditions, and otherwise prohibits the implementation of the designated profile element.

NOTE This is different from conditional because a conditional profile element may be implemented even if the condition is not true.

For any profile element designated as conditional exclusive, the condition shall be defined using one of the mechanisms defined in 7.4.

A CIM based discovery mechanism (see 7.5) shall be defined that enables clients — after having determined that the subject profile is implemented — to determine whether the conditional exclusive profile element is available. The discovery mechanism may be defined indirectly, such that the discovery mechanism for one conditional exclusive profile element by means of conditional dependencies is delegated to that of another profile element; particularly, this is the case with feature implementation conditions (see 7.4.3) and feature discovery (see 7.15.6).

7.3.6 Usage of the "prohibited" requirement level

A subject profile should designate a profile element as prohibited if it prohibits the implementation of the designated profile element. Prohibiting the implementation of certain profile elements might be necessary for example to suppress specific behaviors under certain conditions, or in cases where from a selection of possible variants only one is to be implemented.

7.4 Definition of conditions

This subclause defines mechanisms for the definition of conditions. A condition determines whether a conditional or conditional exclusive profile element must be implemented.

7.4.1 General

As defined in 7.3.4, profiles shall define a condition for any conditional or conditional exclusive elements.

Profiles shall apply only the mechanisms defined in 7.4 defining such conditions. Subclauses 7.4.2 to 7.4.7 define basic types of conditions. Complex conditions may be expressed as combinations of basic conditions using the Boolean operators AND, OR, NOT, XOR and IMPLIES.

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

NOTE 1 Conditions control conditional implementation requirements. Conditions are resolved at implementation time and are complied with by implementers as they implement conditional and conditional exclusive elements in the case where the condition is true. Conditions themselves are not generally directly observable by clients; however, the effect of implementing conditional elements is observable by clients. Discovery mechanisms are CIM based mechanisms that are specifically designed to provide for the run time discovery of optional, conditional or conditional exclusive profile elements; for details, see 7.5.

NOTE 2 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 managed environment addressed by an implementation, not leaving any room for a decision to be made.

7.4.2 Profile implementation condition

A profile may specify a condition based on whether or not a referenced profile is implemented. This kind of condition is called a *profile implementation condition*.

A profile implementation conditional is True if the referenced profile is implemented; otherwise, a profile implementation conditional is False.

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 traversing to CIM_Sensor instances representing attached fan speed sensors 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 conditional 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 server 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.

7.4.3 Feature implementation condition

A profile may specify a condition based on the implementation of a feature (see 7.15). This kind of condition is called a *feature implementation condition*.

A feature implementation condition is True if the feature is implemented as part of a profile implementation, without taking into account the granularity level of the feature; otherwise, a feature implementation condition is False. For details about feature granularity levels, see 7.15.5.

For example, an Example Fan profile might model fan management. This Example Fan profile might define a "FanSpeedSensor" feature. Some elements adapted by the Example Fan profile 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 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 7.4.2 defines it; 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 because 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; for details, see 7.15.6. This generally reduces the number of variations in implementations and therefore the complexity of clients that must accommodate those variations.

7.4.4 Class adaptation implementation condition

A profile may specify a condition based on the implementation of a non-mandatory class adaptation (see 7.13). 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; otherwise, a class adaptation implementation condition is False.

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 the decision to implement the class adaptation is made initially; 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

7.4.5 Instance existence condition

Instance existence conditions are deprecated in favor of the discovery through identified or related adaptation instances (see 7.5.2 and 7.5.3); for the rationale, see the "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; otherwise, the instance existence condition is False. The profile shall define a discovery mechanism for the CIM instance; for details, see 7.5.

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 profile elements (such as adaptations) depends on the existence of CIM instances. Thus a design time decision (such as implementing an 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); consequently, as detailed in Figure 3, the determination of the condition requires the implementer to abstractly anticipate the runtime situation. In other words, the implementer who needs to make a design time decision (for example, implement the adaptation) would have to figure out potential runtime situations (for example, the existence of CIM instances) that are only the result of the implementation; this is considered a cumbersome and potentially error prone exercise.

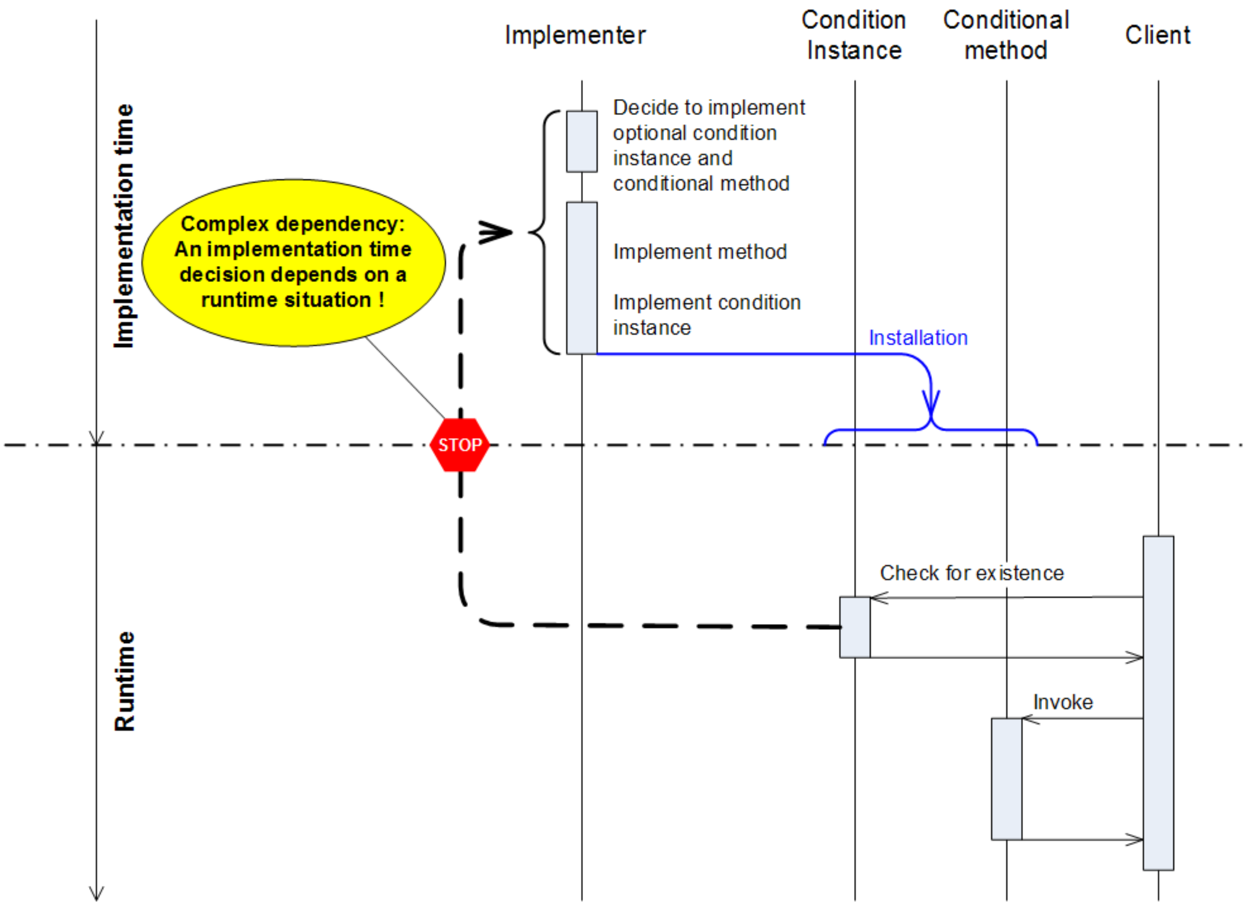

Implementer
Condition Instance
Conditional method
Client
Implementation time
Runtime
Decide to implement optional condition instance and conditional method
Implement method
Implement condition instance
Complex dependency: An implementation time decision depends on a runtime situation !
Installation
STOP
Check for existence
Invoke

Figure 3 – Complexity when an implementation decision depends on a runtime element

Deprecation notice: Instance existence conditions are an unnecessary complication and indirection of the decision process for implementing a conditional or conditional exclusive 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 from specifying the existence of instances as a means for clients to detect the result of design-time decisions. On the contrary, this guide requires profiles to define discovery mechanisms for the run time discovery of conditional or conditional exclusive profile elements (see 7.5). This significantly differs from instance existence conditions insofar as now the design-time decision (for example, the implementation of an optional feature) is made first, and as a consequence the implementation is required to provide discovery elements (such as a specific CIM instance) that indicate the implementation of the conditional or conditional exclusive element to clients.

DEPRECATED

DEPRECATED

7.4.6 Property value condition

Property value conditions are deprecated in favor of discovery through specific property values (see 7.5.4); for the rationale, see the "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; otherwise, the property value condition is False.

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); consequently, similar to the situation detailed in Figure 3, the determination of the condition requires the implementer to abstractly anticipate the runtime situation. In other words, the implementer who needs to make the design-time decision (for example, implement the adaptation) would have to figure out potential runtime situations (for example, 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 or conditional exclusive 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 from specifying property values as a means for clients to detect the result of design time decisions. On the contrary, this guide requires profiles to define discovery mechanisms for the run time discovery of conditional or conditional exclusive profile elements (see 7.5). This significantly differs from property value conditions insofar as now the design time decision (for example, the implementation of an optional class adaptation) is made first, and as a consequence the implementation is required to provide discovery elements (such as a specific property value in a CIM instance) that enable clients to detect the implementation of the conditional or conditional exclusive element.

DEPRECATED

7.4.7 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 managed 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 which the profile is implemented; otherwise, the managed environment condition is False.

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 on the condition that the type of managed environment for which 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 1 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 makes sense only if the implementation has the means to blink the LEDs.

NOTE 2 Of course managed environment conditions are only testable using white box testing where the test code also has access to specific means to test the managed environment condition. Ideally these means would be different from those used by a profile implementation.

7.5 Discovery mechanisms

7.5.1 General

Discovery mechanisms enable clients to discover whether optional, conditional or conditional exclusive profile elements are implemented, or are available in context of other profile elements. A discovery mechanism is a CIM based mechanism that yields a Boolean result.

It is highly recommended that profiles define discovery mechanisms for optional (see 7.3.3), conditional (see 7.3.4) or conditional exclusive (see 7.3.5) profile elements.

7.5.2 Discovery through an identified adaptation instance

For this discovery mechanism the subject profile needs to define an identification for a particular adaptation instance, for example by requiring specific property values. If an instance matching the profile defined identification exists, the discovery mechanism yields True, otherwise False.

An example is an instance of an adaptation of the CIM_RegisteredProfile class that represents the registration of a subject profile (for details on profile registration, see DSP1033). Clients can discover that instance by filtering existing instances for values of the identification properties defined by the subject profile, such as the RegisteredName, RegisteredOrganization and RegisteredVersion properties.

7.5.3 Discovery through a related adaptation instance

For this discovery mechanism the subject profile needs to define an association path from a subject adaptation instance (in context of which the discoverable implementation variant is available) to a related adaptation instance. If the related instance is reachable by traversing the defined association path from the subject adaptation instance, the discovery mechanism yields True, otherwise False. Note that the discoverable implementation variant does not necessarily have to be available in direct context of the subject adaptation instance itself, but instead may apply to elements that are related to the subject adaptation instance.

For example, an Example Port profile could define a PortController adaptation of the CIM_PortController class modeling port controllers, a PortErrorLED adaptation of the CIM_AlarmDevice class modeling a blinkable LED that is capable of signaling an error or a port controller, and an AssociatedLED adaptation of the CIM_AssociatedAlarm association modeling the relationship between a port controller and its error indication LED. Clients can discover whether optional error indication LEDs are installed for a particular port controller by resolving the CIM_AssociatedAlarm association, starting from the PortController instance representing that port controller, for CIM_AlarmDevice instances; if such an instance exists, a client can rely on that optional error indicator LEDs are installed for the port controller.

7.5.4 Implementation discovery through specific property values

This discovery mechanism is applicable for a subject instance itself, or as extension to a discovery mechanisms for an identified instance or a related instance. For such instances, the profile defines specific property values; only if the instance exists and exhibits these specific property values, the discovery mechanism yields True, otherwise it yields False.

For example, an Example Fan profile might define a FanCapabilities adaptation of the CIM_EnabledLogicialElementCapabilities class, and associate that with the Fan adaptation by means of an adaptation of the CIM_ElementCapabilities association. The Example Fan profile might further define that the value of the ElementNameEditSupported property shall have the value True if the modification of the ElementName property in the related Fan instance is implemented. Thus a client can - by inspecting the value of the ElementNameEditSupported property in a FanCapabilities instance associated with a Fan instance – discover that the modification of the ElementName property in the Fan instance is implemented.

7.6 Definition of the profile identification

This subclause defines the elements of a profile identification.

7.6.1 General

A profile shall uniquely identify itself through a registered profile name (see 7.6.2), version (see 7.6.3), and organization (see 7.6.4).

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

7.6.2 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 normal profile text references to other profiles should append the word "profile" to the registered profile name. For example, a profile referencing another profile whose value of the registered profile name attribute is "System Virtualization" would use text such as "If the System Virtualization profile (see DSP1042) is implemented, then …".

NOTE 1 This rule is for references to profiles in normal profile text. It is to be distinguished from the rules for referencing *specification documents* (including profile specification documents), as established by the "Document conventions" of this guide. References to specification documents typically only appear in the "Normative references" and in the "Bibliography" clauses of a profile. For example, when referring to the profile specification document that contains the definition of version 1.0 of the System Virtualization profile and that is titled "System Virtualization Profile", that profile specification document would have to be referenced as DMTF DSP1042, *System Virtualization Profile 1.0* in the "Normative references" clause. It is important to realize that the definition of a profile is different from a document that contains that definition. For example, the definition of the System Virtualization profile could be contained in the document with the number DMTF DSP1042 in the form of a profile specification. Likewise, it could be contained in the document with the number DMTF DSP6042 in the form of a machine readable profile.

NOTE 2 A helpful convention applied by many profile specification documents (and by this guide) when referring to a profile in normal text is appending a phrase such as "(see <docnum>)" after a first reference to a profile within a subclause, where <docnum> is an internal hyperlink. The hyperlink is named as the document number of the referenced document, and links to the entry in the "Normative references" clause that refers to the document that contains the definition of the referenced profile.

7.6.3 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.

DMTF Standard versions of a profile shall specify the major version identifier, the minor version identifier and the update identifier for the registered profile version. Work-in-progress versions of a profile should in addition specify the draft level in order to enable the distinction of implementation of work-in-progress versions from DMTF Standard versions.

7.6.4 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".

7.6.5 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.

7.7 Definition of schema references

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

7.7.1 General

A profile shall reference each schema that defines classes adapted by the profile. Each schema reference shall state the schema name (see 7.7.3), the schema version (see 7.7.2), and the schema organization (see 7.7.4), unless default values apply.

7.7.2 Schema version

The schema version shall be stated with the major version identifier, the minor version identifier and, if needed, the update identifier. The schema version should 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; in other words, while an update identifier identifies the minimally required update version, it shall be interpreted as referring to the latest update version published after the minimally required update version.

7.7.3 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".

7.7.4 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 organization is referenced; if not specified in this case, the default schema organization is "DMTF".

7.7.5 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 7.18 for rules for the specification of experimental content.

7.8 Definition of profile categories

7.8.1 General

As pointed out in 6.2, complex management domains typically can be subdivided into smaller management domains where each subdomain narrows down the area of work or field of activity. In order to reflect this subdivision, two categories of profiles are defined: Autonomous profiles and component profiles.

7.8.2 Autonomous profiles

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.

7.8.3 Component profiles

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 the context of a particular referencing profile, enabling reuse of the component profile in the context of many possible referencing 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 systems. 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.

7.9 Definition of profile relationships

7.9.1 Definition of profile references

7.9.1.1 General

A profile reference is a named profile element within the referencing profile; the rules defined in 7.2.2 apply. A profile reference references a profile by stating the type of the profile reference (see 7.9.1.2), and by identifying the minimally required version of the referenced profile (see 7.9.1.3). In addition, the use of the referenced profile in the context of the referencing profile should be described.

A profile reference establishes either profile derivation or a profile usage.

Profile derivation establishes another profile as a base profile of the subject profile; profile derivation is detailed in 7.9.2.

A profile usage establishes a use of the referenced profile within the context of the referencing profile. It is possible that a subject profile defines multiple usages of a particular profile; in this case the subject profile references that profile multiple times, each time for a separate use. 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.

Scoping is a refinement of a profile usage that in addition requires the definition of specific adaptations and dependencies between them in the referencing profile as well as in the referenced profile; for details, see 7.9.3.

A profile shall not reference its previous versions.

The definition of cyclic profile references is allowed for profile usages; however, it is prohibited for profile derivation. Additional restrictions apply in context of cyclic references between profiles. For example, it is not possible to define cyclic relationships between adaptations; for details, see 7.13.2.1.

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

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

7.9.1.2 Types of profile references

The types of profile references are defined as follows:

- **Derivation**
  A derivation profile reference indicates that the definitions of the referenced profile are the base for the referencing profile, as detailed in 7.9.2. 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. As required in 7.9.2, at most one direct base profile shall be established per subject profile.

All subsequent types of profile references establish profile usages:

- **Mandatory**
  A mandatory profile usage indicates that the definitions of the referenced profile apply in the context established by the referencing profile. In this case, the referenced profile is termed a mandatory profile of the referencing profile.

- **Conditional**
  A conditional profile usage indicates that the definitions of the referenced profile under specified conditions apply in the context of the referencing profile. In this case, the referenced profile is termed a conditional profile of the referencing profile.

- **Conditional exclusive**
  A conditional exclusive profile usage indicates that the definitions of the referenced profile under specified conditions apply in the context of the referencing profile, and shall not apply if the specified conditions do not apply. In this case, the referenced profile is termed a conditional exclusive profile of the referencing profile.

- **Optional**
  An optional profile usage indicates that the definitions of the referenced profile optionally apply in the 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.

A referencing profile shall indicate the type of profile reference by using the respective keyword, as designated in **bold face** in the previous list.

As a consequence of a profile reference, the definitions and requirements of the referenced profiles become part of the set of definitions and requirements that are effective for the referencing profile; however, this applies in different ways for profile derivation as opposed to profile usages. The process of how to determine the definitions and requirements that effectively apply for an implementation implementing a set of profiles are detailed in clause 9.

7.9.1.3 Identification of the minimally required version of a referenced profile

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

- the registered profile name of the referenced profile (see 7.6.2)
- the major version identifier, the minor version identifier and optionally the update identifier of the registered profile version of the referenced profile (see 7.6.3). The update identifier should only be used in cases where dependencies on the referenced update version exist that are not already addressed by the minor version.
- the registered organization (see 7.6.4) of the referenced profile

Regardless of whether an update identifier is stated, the latest published update version with the stated major and minor version identifier is referenced; in other words, while an update identifier identifies the minimally required update version, it shall be interpreted as referring to the latest update version published after the minimally required update version. For further details, see DSP4004.

7.9.1.4 Prohibition of the relaxation of requirements

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 remove any constraints established by its referenced profiles.

7.9.1.5 Rules for the repetition of content from referenced profiles

A referencing profile shall not repeat content of its referenced profiles unless it establishes additional constraints. Even in this case repetitions should be avoided unless necessary to establish a context for the additional constraints.

NOTE For rules on the repetition of schema content as part of property requirements, see 7.13.2.8.3.

7.9.1.6 Rules for derived adaptations

A profile may define adaptations based on adaptations defined in referenced profiles; for details, see 7.13.2.1 and 7.13.2.4.

In this case the profile relationships to each profile defining one or more base adaptations shall be defined in compliance with the following rules:

- If mandatory base adaptations are defined, the relationship to each referenced profile defining a mandatory base adaptation shall be mandatory or derivation.
- If conditional base adaptations are defined, the relationship to each referenced profile defining a conditional base adaptation shall be mandatory, derivation, conditional, or conditional exclusive. In the case of conditional or conditional exclusive, the condition shall be at least the conjunction of all individual conditions, or stronger.

7.9.2 Definition of profile derivation

7.9.2.1 General

Subclause 7.9.2 defines rules that ensure that a client that exploits the management interface defined by a base profile can likewise interact through that management interface with profile implementations of the base profile or with those of derived profiles.

DEPRECATED

Version 1.0 of this guide defined the term *profile specialization*. This term 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

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

New derived profiles written in conformance to this guide shall be based on exactly one direct base profile. Minor revisions of existing profiles written in conformance with version 1.0 of this guide that define more than base profile in the original profile may retain defining more than one direct base profile.

DEPRECATED

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

DEPRECATED

In this guide, when referring to more than one base profile, this means the direct base profile and possible indirect base profiles. This is because 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
- features
- profile references
- registry references
- adaptations (including their property requirements, method requirements, operation requirements and metric 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 in 7.9.2 constrain the possible choices of a derived profile.

NOTE Special implementation requirements apply for derived profiles. For example, all implementation requirements defined by a derived profile need to be merged with those of its base profiles; for details, see clause 9.

7.9.2.2 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 becomes 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 a profile implementation of a derived profile to the same extent as with a profile implementation of the base profile.

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

7.9.2.3 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.

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 are allowed to reduce that to any subset, such as "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, there are rules for extending or reducing the value set for input/output parameters and return values in a derived profile; see 7.13.3.2.2. Likewise, this applies to properties that are readable and writable.

NOTE A profile implementation of a derived profile is required to satisfy the requirements of all its (direct and indirect) base profiles. Thus, a client written against the management interface defined by a base profile also works with a profile implementation of a derived profile. Implementation requirements are detailed in clause 9.

7.9.2.4 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, mandatory or prohibited, 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; in this case 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 to be present also in the case where the Y feature is implemented.

7.9.2.5 Definition of schema references

A derived profile shall reference each schema that defines classes adapted by the profile; see 7.7 for a definition of the elements of schema references.

A derived profile may introduce new schema references.

The version of a referenced schema in a derived profile shall not be less recent than the most recent version of that schema in any base profile. A derived profile may refine a schema reference of a base profile by requiring a more recent version of the referenced schema.

7.9.2.6 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 7.13.2.1 apply.

The central class adaptation of a derived profile shall be based on the central class adaptation of its direct base profile. For the adapted class and for other base adaptations the provisions of 7.13.2.1 apply.

7.9.2.7 Propagation of profile references

A derived profile inherits all profile references (see 7.9.1) defined by its (direct or indirect) base profiles; this also applies to the names of the profile references.

A derived profile may introduce new profile references.

A derived profile may override a profile reference made in a base profile with a profile reference that references a profile derived from the profile referenced by the base profile. An overriding profile reference defined in a derived profile shall state the same profile reference name as that used by the profile reference defined in the base profile; in effect, the use of the same profile reference name establishes the override.

7.9.2.8 Propagation of registry references

A derived profile inherits all registry references (see 7.12) defined by its (direct or indirect) base profiles; this also applies to the names of the registry references.

A derived profile may introduce new registry references.

A derived profile may override registry references made in base profiles with registry references that reference compatible registries. New minor or update versions of the originally referenced registry version are always compatible. New major versions of the originally referenced registry version and different registries are compatible to the originally referenced registry version if all registry elements required by the base profile(s) are compatibly defined in that registry version. An overriding registry reference defined in a derived profile shall state the same registry reference name as that used by the registry reference defined in the base profile; in effect, the use of the same registry reference name establishes the override.

7.9.2.9 Propagation of features

A derived profile inherits all features (see 7.15) defined by its (direct or indirect) base profiles; this also applies to the names of the features.

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.

A derived profile may refine features defined in base profiles. For a refined feature it is required that the set of definitions conditional on the refined feature is a superset of the set of definitions conditional on the original feature, that is, the refined feature requires at least the definitions of the original feature, but may require more definitions. An overriding feature defined in a derived profile shall state the same name as that used by the feature defined in the base profile; in effect, the use of the same name establishes the override.

7.9.2.10 Propagation of adaptations

A derived profile inherits adaptations (see 7.13) defined by its (direct or indirect) base profiles in the following two cases:

> **Case A** : The derived profile defines a new adaptation that is based on one or more adaptations defined in its base profiles. In this case, the rules for basing an adaptation on other adaptations as defined in 7.13.2.1 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. A not apparent source for naming conflicts is 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.

7.9.2.11 Propagation of state descriptions and use cases

A derived profile inherits all state descriptions (see 7.16.2) and use cases (see 7.16) defined by its (direct or indirect) base profiles. A derived profile may introduce new state descriptions and use cases.

A derived profile may refine and extend state descriptions and use cases defined in base profiles. A refinement replaces the use of some adaptations defined in base profiles in with that of respective derived adaptations defined in the subject profile. An extension of a use case adds additional steps. An extension of a state description adds additional adaptation instances. A refinement or extension of a state description or use case defined in a derived profile shall state the same name as that used by the state description or use case defined in the base profile; in effect, the use of the same name establishes the refinement or extension.

7.9.3 Definition of scoping relationships

7.9.3.1 General

Scoping is a refinement of profile usage (see 7.9.1) that optimizes the conformance advertisement of component profile implementations by reducing the number of required CIM_ElementConformsToProfile association instances; for details, see 7.14 and DSP1033.

Four elements contribute to defining a scoping relationship:

- The central class adaptation (see 7.9.3.2) defined by the used profile
- The scoping class adaptation (see 7.9.3.3) defined by the used profile

- The scoping path (see 7.9.3.4) defined by the used profile
- The central class adaptation (see 7.9.3.2) defined by the referencing profile

A scoping relationship is established with a profile usage if the central class adaptation defined by the referencing profile is based on (see 7.13.2.1) the scoping class adaptation defined by the used profile.

For example, an Example Fan profile might define a FanSystem adaptation of the CIM_System class as its scoping class adaptation, and an Example Computer System profile might define its ComputerSystem adaptation of the CIM_ComputerSystem class as the central class adaptation, and base it on the FanSystem adaptation of the Example Fan profile. In this case the Example Computer System profile defines a scoping relationship to the Example Fan profile, because the central class adaptation of the referencing profile is based on the scoping class adaptation of the used profile.

Note that not every profile usage implies a scoping relationship; a scoping relationship is only defined if the central class adaptation of the referencing profile is based on the scoping class adaptation of the used profile. For example, the Example Fan profile might reference an Example Sensors profile that defines a SensorSystem adaptation of the CIM_System class as its scoping class adaptation; in this case the Example Fan profile does not (and cannot for class compatibility reasons; see 7.13.2.1) define its central class adaptation based on the scoping class adaptation of the Example Sensors profile.

7.9.3.2 Central class adaptation

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

For requirements relating to profile registration, see 7.14.

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.

7.9.3.3 Scoping class adaptation

A component profile (see 7.8.3) shall designate exactly one mandatory class adaptation as the scoping class adaptation. In this case, the scoping class adaptation shall be different from the designated central class adaptation (see 7.9.3.2).

An autonomous profile (see 7.8.2) shall either not designate a scoping class adaptation, or shall designate the same class adaptation as both the central class adaptation (see 7.9.3.2) and the scoping class adaptation.

For requirements relating to profile registration, see 7.14.

The scoping class adaptation provides an external attach point for scoping profiles. A scoping profile may connect to that attach point by defining its central class adaptation based on the scoping class adaptation defined in used profiles.

7.9.3.4 Scoping path

A scoping path is an association traversal path defined by the subject profile connecting 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.

7.9.3.5 Examples of scoping relationships

- Autonomous profile with optional component profiles

Embedded control systems optionally include management interfaces for elements such as fans or power supplies. In this case, the primary management interface addressing the core functionality of the control systems would be defined in the autonomous profile, whereas the secondary management interfaces addressing the functionality of the fan and power supply elements would be defined in separate component profiles. This is shown in Figure 4.

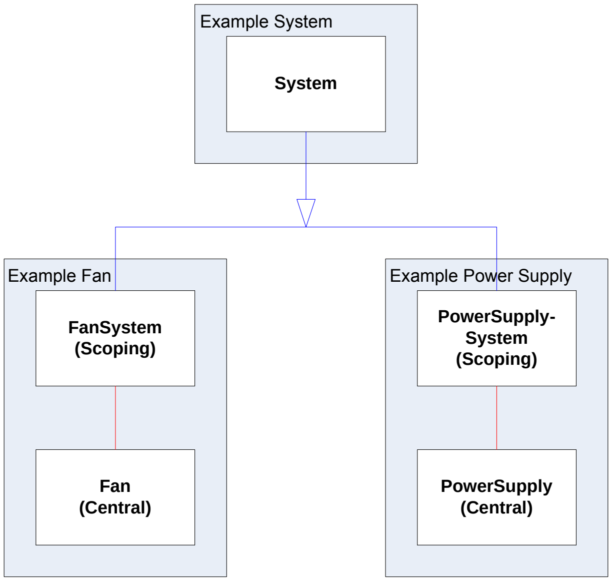


Figure 4 – Autonomous profile with optional component profiles

- Multiple autonomous profiles sharing component profiles

  Disk arrays and volume managers provide similar RAID virtualization capabilities from a device of host-resident software. In this case, a RAID virtualization component profile could be referenced (shared) by an Array (external virtualization hardware) autonomous profile, and by a Volume Manager (host-resident virtualization software) autonomous profile.

- 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 an autonomous profile in the scoping hierarchy.

- Scoping between component profiles

  Figure 5 shows two variants of an Example Fan profile referencing an Example Sensors profile:

  – The left side of Figure 5 shows the example with a scoping relationship established by an autonomous Example System profile for both an Example Fan and an Example Sensors profile by basing the Example System profile's System adaptation on both the FanSystem adaptation of the Example Fan profile and the SensorSystem adaptation of the Example Sensors profile.

  – The right side of Figure 5 shows a variant of this example with the scoping relationship for the Example Sensors profile established by the Example Fan profile; in this case the Example Fan profile bases its (central) Fan adaptation on the (scoping) SensoredElement adaptation of the Example Sensors profile, thereby establishing a scoping relationship. Note that the SensoredElement adaptation adapts the CIM_ManagedSystemElement class. That way any profile adapting the CIM_ManagedSystemElement class (or a subclass thereof) as its central class adaptation could define a scoping relationship to the Example Sensors profile.

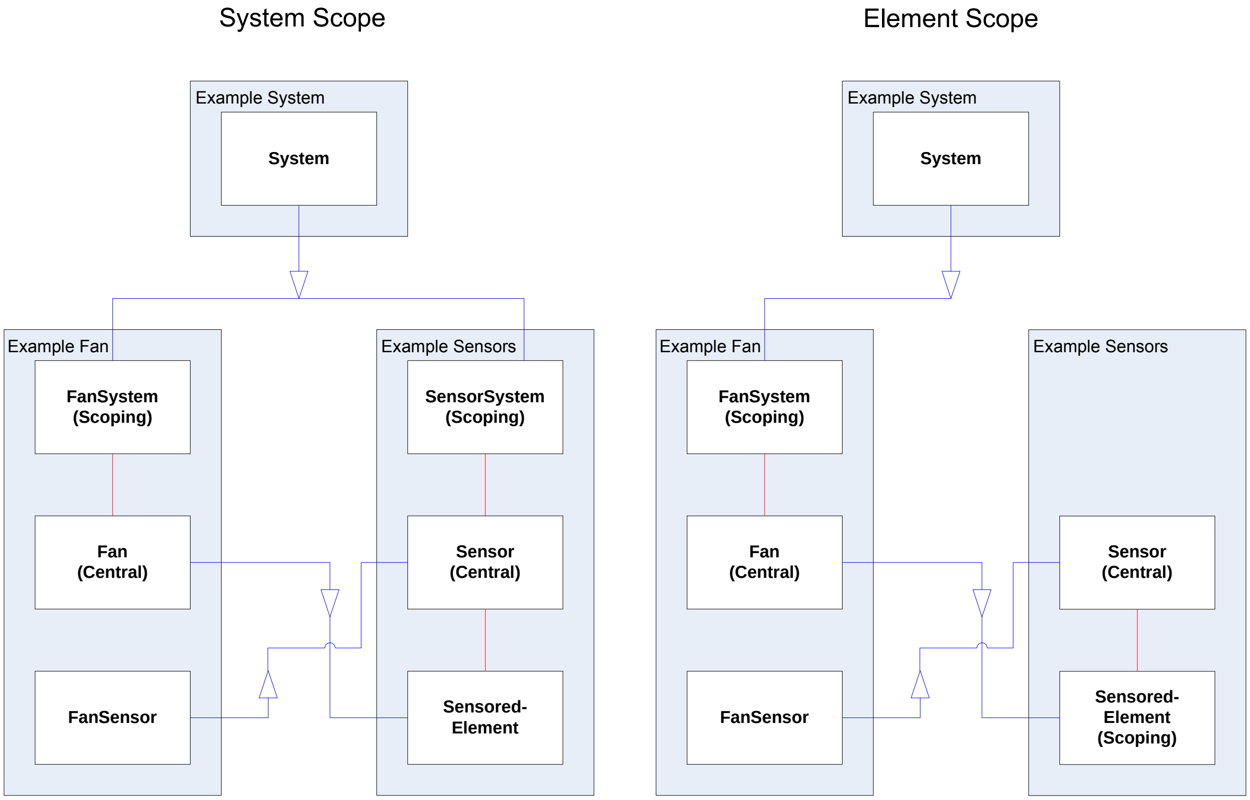


Figure 5 – Two variants of a component profile using another component profile

Note that the right variant shown in Figure 5 would require the central class profile advertisement methodology as defined in the Profile Registration profile (see DSP1033) to be implemented for the Example Fan profile because version 1.0 of the Profile Registration profile does not allow the scoping class profile advertisement methodology span two or more levels of profiles.

7.10 Definition of abstract and concrete profiles

7.10.1 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 directly; instead, the definitions and requirements of an abstract profile are propagated into derived profiles (see 7.9.2) and apply for profile implementations implementing concrete derived profiles.

An abstract profile may define class adaptations of concrete classes and/or abstract classes.

An abstract profile may define concrete class adaptations and/or abstract class adaptations.

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.

Profiles may define a profile usage relationship to abstract profiles. For example, a profile addressing the management domain of virtual computer system could define a profile usage of an abstract profile addressing the management domain of allocating resources to consumers.

7.10.2 Concrete profile

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

Specific requirements for the definition of adaptations of abstract classes apply; see 7.13.5.

Furthermore, 7.14 defines requirements for concrete profiles related to profile registration.

7.11 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 define the functionality of respective managed objects to the extent exposed by the model defined by the profile such that an implementer who implements the profile for a particular type of managed environment is enabled to realize the profile defined mappings (see 6.6.1).

In some cases it may be sufficient to refer to respective definitions in the schema definition of adapted classes. However, generally profiles adapt generic classes to model a more specific managed object type than that described in the schema definition of 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 is 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 an identifier that uniquely identifies the file in the scope of a file system. Files may be referenced by one or more directories; each such file reference defines a file name that shall be unique within the referencing directory.

A *directory* is a special kind of file that contains a list of references to files; each list entry references one file. A directory shall assign a name to each referenced file that is unique in scope of the directory.

In this example the management domain definition shown in Table 1 would enable a profile 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.

7.12 Definition of registry references

Profiles may reference message registries and metric registries.

Message registries are registries that conform to DSP0228 and contain message definitions.

Metric registries are registries that conform to DSP8020 and contain metric definitions.

A registry reference is a named profile element (see 7.2.2) that references a registry by stating the type of the referenced registry and by identifying the minimally required version of the referenced registry. A subject profile defining registry references should provide a description that details the use of each referenced registry within the subject profile.

A registry reference shall be assigned a name as defined in 7.2.2.

NOTE The use of a local name for registry references provides for the possibility of overrides if subsequent versions of a profile need to refer to a different registry that compatibly supersedes the originally referenced registry; see 7.9.2.8. Furthermore, the local name is used to identify the registry when referencing elements defined within the registry.

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

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

- the unique identifier of the registry as assigned by the owning organization. For registries conforming to DSP0228 or DSP8020, this is the value of the ID attribute; the fully qualified XPATH location of the ID attribute in both types of registry is /REGISTRY/REGISTRY_DECLARATION/IDENTIFICATION/@ID.
- the major version identifier, the minor version identifier, and optionally the update identifier of the registry. The update identifier should only be used in cases where dependencies on the update version exist that are not already addressed by the minor version. Regardless of whether an update identifier is stated, the latest published update version with the stated major and minor version identifier is referenced; in other words, while an update identifier identifies the minimally required update version, it shall be interpreted as referring to the latest update version published after the minimally required update version. For further details, see DSP4004.
- the organization that owns the registry

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

As part of their other definitions, profiles may refer to metric definitions defined in metric registries.

7.13 Definition of class adaptations

7.13.1 General

A class adaptation is a named profile element; the rules defined in 7.2.2 apply. Class adaptations may be referred to simply as *adaptations*.

An adaptation defines the use of a class defined in a schema for a particular purpose.

In addition to *adapting* a schema defined class, an adaptation may further be *based on* one or more other adaptations. The subject profile may establish further constraints for an adaptation beyond those established by the schema definition of the adapted class, or by referenced adaptations.

DEPRECATED

Profiles that were created in conformance with version 1.0 of this guide did not define adaptations, but so called "*profile classes*" (sometimes also called "profiled class", "supported class" or just "class"). The concept of "profile classes" obliterated the distinction between the schema definition of a class, and the profile defined use of the class. The semantics of "profile classes" can viewed as a subset of the semantics of adaptations; for example, "profile classes" lack the ability to be based on each other. A "profile class" used the name of the adapted schema class; that name could be suffixed with an optional modifier in order to resolve name clashes.

Minor revisions of profiles specified in compliance with version 1.0 of this guide may continue using the following naming convention for adaptations (stated in ABNF):

```
ProfileClassName = SchemaClassName [ "(" Modifier ")" ]
```

`SchemaClassName` is the name of the class defined in the schema. `Modifier` is a short descriptor that describes the use of the adapted class in the 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 "profile classes" in minor revisions of existing profiles. Newly introduced adaptations in minor revisions shall not apply this naming convention.

DEPRECATED

7.13.2 Requirements for definitions of all kinds of adaptations

This subclause defines requirements for definitions of all kinds of adaptations: Adaptations of ordinary classes, adaptations of association classes, and adaptations of indication classes.

7.13.2.1 Adapted class and base adaptations

An adaptation adapts a class defined in a schema for a particular purpose; this class is called the adapted class.

In addition, an adaptation may be based on zero or more other adaptations; these adaptations are called base adaptations.

For a particular adaptation, the following rules apply:

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

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

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

  An adaptation may reference one 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 other 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 a profile 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 profile implementation of a new component profile to a previously existing implementation of a set of profiles, where the new component profile is not referenced.

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

Figure 6 shows an example that illustrates how the rules defined in this subclause establish limitations for the selection of base adaptations or of adaptable classes, after an initial choice is made.

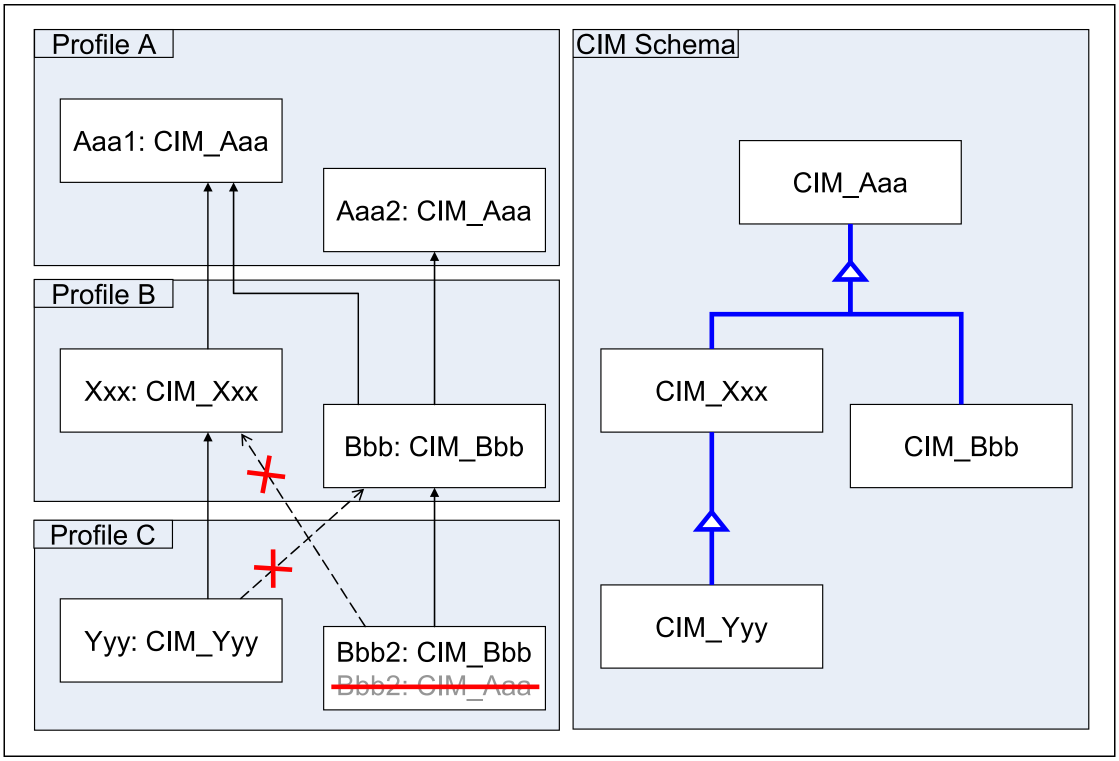


Figure 6 – Class adaptation reference example

In the example shown in Figure 6, 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 adaptation of a class that was marked as deprecated in a later version of the schema.

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

- All definitions and requirements defined by base adaptations are propagated into the adaptation.
- The potential set of instances of an adaptation shall be a subset of the potential set of instances of each of its base adaptations. For example, if the VirtualSystem adaptation defined by an Example Virtual System profile is based on the ComputerSystem adaptation of an Example Computer System profile, then the potential set of instances of the VirtualSystem adaptation is required to be a subset of the potential set of instances of the ComputerSystem adaptation.

DMTF collaboration structure diagrams (see 8.3.4) are specifically tailored to graphically depict the dependencies introduced by basing adaptations on other adaptations.

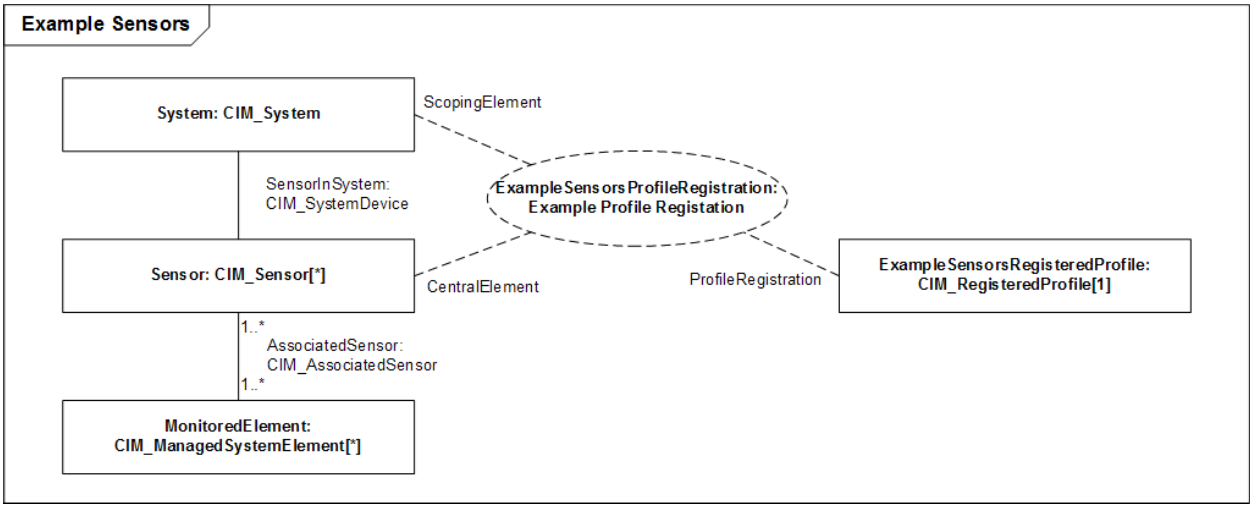


Figure 7 – DMTF collaboration structure diagram of an Example Sensors profile

Figure 7 shows the DMTF collaboration structure diagram of an Example Sensors profile; for details about DMTF collaboration structure diagrams, see 8.3.4.

In Figure 7, the dashed oval labeled "ExampleSensorsProfileRegistration: Example Profile Registration" represents the Example Sensors profile's reference to the Example Profile Registration profile. The solid rectangle labeled "Sensor: CIM_Sensor" represents the Example Sensors profile'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 Example Profile Registration profile. Likewise, the System adaptation of the Example Sensors profile is based on the ScopingElement adaptation of the Example Profile Registration profile, and the ExampleSensorsRegisteredProfile adaptation of the Example Sensors profile is based on the RegisteredProfile adaptation of the Example Profile Registration profile.

The capability of basing adaptations on other adaptations enables encapsulation, resulting in simplified modeling approaches. For example, in Figure 7 an adaptation of the CIM_ElementConformsToProfile association is not shown. Instead, it is assumed that a respective association adaptation is defined by the Example Profile Registration profile. That way, the different approaches to modeling the functionality related to profile registration is exclusively defined in the Example Profile Registration profile, and there is no need to refine that adaptation in the Example Sensors profile.

Furthermore, the capability of basing adaptations defined in one profile on adaptations defined in referenced profiles provides for a much finer granularity of profile dependencies: With this approach requirements are introduced at the level of adaptations rather than at the level of profiles. For example, the approach of basing the central and scoping adaptations on respective adaptations of the Example

Profile Registration Profile as shown in Figure 7 is much stricter than that of only referencing the Example Profile Registration Profile as a mandatory profile.

7.13.2.2 Management domain context of class adaptations

For each adaptation it defines, the subject profile shall state the managed object type from the management domain (or the aspect of a managed object type) that is modeled by the adaptation. See 7.11 for requirements on defining the management domain and its managed object types.

NOTE Elements from the CIM infrastructure can also be described by managed object types, such as, for example, registered profiles or indication filters. While without CIM these elements would not exist as managed objects in a managed environment (unlike, for example, computer systems or file systems), they are part of the managed environment if CIM is applied for defining and realizing the management infrastructure, and are modeled by adaptations of CIM classes. For example, an Example Profile Registration profile might model a RegisteredProfile adaptation of the CIM_RegisteredProfile class modeling the managed object type "registered profile", or an Example Indications profile might model an IndicationFilter adaptation of the CIM_IndicationFilter class modeling the managed object type "indication filter".

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

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

7.13.2.3 Requirement level

For each adaptation it defines, the subject profile shall designate a requirement level that determines the requirement for implementing the adaptation as part of the profile implementation of the subject profile.

7.13.2.4 Individual requirement levels of base adaptations

If an adaptation is based on other adaptations (see 7.13.2.1), then each such relationship shall be designated with a separate requirement level that determines the requirement for implementing the base adaptation as part of implementing the subject adaptation.

NOTE The typical requirement level for a base adaptation is mandatory. In some cases a requirement level of conditional/conditional exclusive for a feature is a favorable alternative. As an example, consider the case in which the subject profile defines an optional Metrics feature. In this case, some adaptations of the subject profile would typically be based on adaptations defined in the Base Metrics profile, but only if the optional Metrics feature of the subject profile is implemented.

7.13.2.5 Implementation type

Each adaptation shall be designated with an implementation type that details how the adaptation is to be implemented.

The following implementation types are possible:

instantiated: indicates that the adaptation is to be implemented such that instances of the adaptation are instantiated on their own, i.e. they can be referenced with an instance path by a client.

embedded: indicates that the adaptation is to be implemented such that instances of the adaptation are embedded into an embedding element; they cannot directly be referenced with an instance path by a client.

abstract: indicates that the implementation type of the adaptation is defined by its derived adaptations. Profiles shall assign the abstract implementation type if the functionality defined by the adaptation is not independently required for a functioning profile implementation, but instead is designed to be refined by other adaptations (defined in the same, or in other profiles) that define the abstract class adaptation as a base adaptation (for details, see 7.13.2.1). Insofar, the use of the abstract implementation type delegates the selection of an implementation type to adaptations based on the abstract class adaptation.

indication: indicates that the adaptation is to be implemented such that instances of the adaptation are embedded as elements in indication delivery operations. The "indication" implementation type is only applicable for adaptations of classes that have effective qualifier values of Indication=True and Exception=False.

exception: indicates that the adaptation is to be implemented such that instances of the adaptation are embedded into operation exceptions (typically delivered as fault responses of operations). The "exception" implementation type is only applicable for adaptations of classes that have effective qualifier values of Indication=True and Exception=True.

DEPRECATED

Profiles that were created in conformance with version 1.0 of this guide did not designate adaptations with an implementation type. Minor revisions of profiles specified in compliance with version 1.0 of this guide may continue not designating an implementation type to the adaptations they define. In this case, a default implementation type shall be assumed, as follows:

- For adaptations of classes that have effective qualifier values of Indication=True and Exception=False, the default implementation type is "indication".
- For adaptations of classes that have effective qualifier values of Indication=True and Exception=True, the default implementation type is "exception".
- For all other adaptations, the default implementation type is "instantiated".

DEPRECATED

7.13.2.6 Designation of base adaptation candidates

A profile may designate individual adaptations as base adaptation candidates. The purpose of this designation is conveying to authors of referencing profiles that — from the perspective of the defining profile — the designated adaptation models a functional element with the intention to be refined by means of defining derived adaptations in referencing profiles.

NOTE Formally, any adaptation defined in a profile can be used as a base adaptation; however, the specific designation of an adaptation as a base adaptation candidate is intended to serve as a hint to authors of referencing profiles for considering the definition of a derived adaptation.

7.13.2.7 Use of the value Null as property or parameter value

DSP0223 requires that on method invocation values are provided for all input parameters, and on method return values are returned for all output parameters and for the method return value. However, unless otherwise required by profiles and/or the schema, Null is a legal value. DSP0004 states that the special value Null indicates the absence of a value. Profiles should avoid assigning the value Null a semantic other than that defined in DSP0004. Profiles should specify the implementation behavior in the case of the absence of an input parameter value (that is, an input value Null). Profiles should specify how the absence of an output parameter value or of a method return value (that is, an output value Null) is to be interpreted. This applies likewise to property values in adaptation instances that are used as input or output value for parameters of methods or operations, or as method return values.

7.13.2.8 Definition of property requirements

7.13.2.8.1 General

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

7.13.2.8.2 Requirement level

Each property requirement shall be designated with a "presentation" requirement level that determines the requirement for implementing the property as part implementing the adaptation for the purpose of presenting information.

In addition, for adaptations with the "instantiated" implementation type (see 7.13.2.5) that a profile defines as creatable and/or modifiable by clients, separate requirement levels for specific property values may be specified:

- An "initialization" requirement level that determines if the specific value shall be implemented as a property initialization value; for details, see 7.13.2.11.2.
- A "modification" requirement level that determines if the specific value shall be implemented as a property modification value; for details, see 7.13.2.11.3.

7.13.2.8.3 Rules for the repetition of schema requirements

In adaptations mandatory property requirements shall be defined for all key properties and for all properties for which the Required qualifier has an effective value of True, unless respective property requirements are already stated by a base adaptation.

NOTE This requirement aims at relieving profile consumers from analyzing the schema for respective requirements.

Otherwise, a subject profile should not replicate requirements from the schema or from base profiles unless needed for establishing additional requirements of the subject profile.

7.13.2.8.4 Requirements for the specification of property constraints

The base set of permissible property values is defined by schema definition of the adapted class and/or its superclasses; as a matter of principle, schema definitions cannot be extended by profiles.

A profile may specify constraints and requirements 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, its superclasses and any base adaptation.

In other words, profiles shall not specify property requirements that extend the set of permissible property values as constrained in base adaptations, but may specify property requirements that further constrain the set of permissible property values.

In addition, for adaptations with the "instantiated" implementation type (see 7.13.2.5), separate value constraints may be specified for the presentation, the initialization and the modification of the property value; however, the value constraints for the initialization and modification shall be within those defined for the presentation.

The schema definition of the adapted class, its superclasses, or any base adaptation may specify rules that prohibit or establish limitations for the definition of such constraints in general, or under certain conditions.

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.

7.13.2.8.5 Management domain context of properties

As part of every property requirement, the profile shall specify the aspect of managed objects that represented by adaptation instances and is reflected by the property, unless that aspect 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.

7.13.2.9 Default values for properties, parameters and method return values

A profile may specify a default value for a property, parameter or method return value. Profile specified default output values apply in the case where a more specific value is indiscernible by the profile implementation. For example, a profile could define the empty string "" as a default value for the ElementName property that is required by the schema to have a non-Null value. In this case that value would have to be returned in the case where a profile 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. In the schema default values can only be defined for properties and are considered initialization constraints; initialization constraints determine the initial value of the property in new instances; see also 7.13.3.3.3.

7.13.2.10 Value constraints for properties, parameters and method return values

7.13.2.10.1 General

Profiles may define value constraints for properties, parameters and method return values using various mechanisms such as restricting a set of distinct values of numeric or string type in a value map, restricting a numeric value range, restricting bits in a bit map or constraints based on logical expressions of other constraints.

If a profile defines value constraints, these should be defined allowing for adequate margin with respect to the implementations ability to represent (aspects of) managed objects by adaptation instances (see 7.13.2.8.5), and with respect to represent the outcome of a method execution in the method result (see 7.13.3.2.2 and 7.13.3.2.3).

Value constraint do not imply value requirements; in other words, it is not required that all the values from the value set determined by the conjunction of the all value constraints are implemented. However, for input values, specific input value requirements may be specified (see 7.13.2.11).

NOTE This guide also establishes specific conventions for the specification of value constraints in profile specifications; for details, see 10.2.4.

7.13.2.10.2 Value constraints for reference values

Profiles may define constraints as part of property requirements for reference properties in association adaptations, and as part of method requirement for reference parameters and reference method return values, as follows:

- The constraint shall state the adaptation that the reference property refers to. It is required that the referenced adaptation is defined in the subject profile.
- The referenced adaptation shall be compatible with the class that is referenced by the reference property, parameter or return value in the adapted class; for details, see 7.13.2.1.

- 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 and its adapted class.

As a consequence of the first rule, it is not possible that a subject profile can define an association adaptation that references an adaptation defined in a referencing profile because the referencing profile and its adaptation are not known in the subject profile. This situation can be solved by defining the associated adaptation directly in the subject profile, and base the adaptation in the referencing profile on the new adaptation in the referenced profile. In most cases the adaptation in the subject profile can be stated as a trivial class adaptation (see 7.13.6) which causes only minimal modeling effort. The advantage of this approach is that the adaptation dependencies are explicitly defined and it is not left to the implementer to figure out which adaptation in a referenced profile actually referenced.

For example, consider an Example Fan profile modeling a relationship between a fan and the system that contains the fan by means of the CIM_SystemDevice association. That profile would model a Fan adaptation of the CIM_Fan class, a (trivial) FanSystem adaptation of the CIM_System class, and a FanInSystem adaptation of the CIM_SystemDevice association that references the Fan and the FanSystem adaptations.

NOTE Version 1.0 of this guide does not clearly separate adaptations (which were called "profile classes" – see 7.13.1) and CIM classes. DMTF profile class diagrams in component profiles conforming to version 1.0 of this guide frequently depict "profile classes" from a referencing profile and annotate it with the phrase "See referencing profile". Implementers of such profiles in context of a particular referencing profile now need to determine which "profile class" in the referencing profile is actually referenced. This is a trivial task if only one "profile class" for the respective CIM class is defined in the referencing profile, but causes ambiguities if more than one "profile class" of that CIM class is defined, and the association reference is not further constrained to reference a particular "profile class".

7.13.2.10.3 Value constrains through format specifications

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.

7.13.2.10.4 Property non-Null value constraint implied by the requirement level

If a property is required by a subject profile with either the mandatory requirement level, or with the conditional or conditional exclusive requirement level and the condition being True, the value Null is not admissible for the property (see 9.3.2).

Profiles may exempt this rule and allow Null as an admissible value; however, such exemptions should be specified separately for each property where the value Null is admissible.

A respective value constraint is not implied for the use of Null as an input value; however, specific input value requirements may be defined (see 7.13.2.11).

7.13.2.11 Input value requirements

7.13.2.11.1 General

Input value requirements are requirements for the implementation of particular input values.

An input value requirement requires that the input value must be implemented, that is, be accepted when provided as input, and not be rejected for the reason of not being implemented; however, a rejection for other reasons is not prohibited. Input value requirements may be specified for specific values of method input parameters, and — with respect to the initialization or modification of property values — for specific property values as part of property requirements in adaptations.

NOTE Value requirements for output values can only be specified by means of value constraints (see 7.13.2.10). Recall that property values are required to represent the state of the managed environment represented by the adaptation instance (see 7.13.2.8.5), and that method return values and method output parameter values are required to represent the outcome of the method execution (see 7.13.3.2.2 and 7.13.3.2.3).

7.13.2.11.2 Property initialization value requirement

Property initialization value requirements are input value requirements that may be specified with property requirements in the definition of adaptations with an implementation type (see 7.13.2.5) of "instantiated". Property initialization input value requirements shall not be specified in the definition of adaptations with other implementation types.

Each property initialization value requirement shall be designated with a requirement level that determines the requirement for implementing the value as property initialization value.

A property initialization value requirement states that a specific input value for a property shall be implemented, that is, be accepted when provided through any operation or method that creates instances of the adaptation (such as the CreateInstance( ) operation defined in DSP0223, or as methods that take an embedded adaptation instance as input). A property initialization value requirement is only applicable if such operations or methods are implemented.

Implementing a property initialization value does not preclude its rejection for reasons other than not being implemented, such as that the state of the managed environment does not currently allow the instance creation request to be executed with the given input instance.

Property initialization value requirements shall only be specified for values that are within the value constraints established for the property (see 7.13.2.10). In addition, creation methods or operations may define separate constraints that limit their specific sets of acceptable values beyond those defined by property constraints.

If for a possible value no property initialization value requirement is specified, the implementation may either accept or reject that value when provided as initialization value.

The semantics of the creation operation or method may define how initialization values are processed. Defining semantics includes the possibility that an initialization value is only considered a hint, such that the value resulting from the instance creation differs from the provided initialization value. If no specific semantics are defined, the default shall be that the initialization value is carried over unmodified into the new instance.

7.13.2.11.3 Property modification value requirement

Property modification value requirements are input value requirements that may be specified with property requirements in the definition of adaptations with an implementation type (see 7.13.2.5) of "instantiated". Property modification value requirements shall not be specified in the definition of adaptations with other implementation types.

Each property modification value requirement shall be designated with a requirement level that determines the requirement for implementing the value as property modification value.

A property modification value requirement states that a specific value for a property must be implemented, that is, be accepted when provided through any operation or method that modifies instances of the adaptation (such as the ModifyInstance( ) operation defined in DSP0223, or as methods that take an embedded adaptation instance as input). A property modification value requirement is only applicable if such operations or methods are implemented.

Implementing a property modification value does not preclude its rejection for reasons other than not being implemented, such as that the state of the managed environment does not currently allow the instance modification request to be executed with the given input instance.

Property modification value requirements shall only be specified for values that are within the value constraints established for the property (see 7.13.2.10). In addition, modification methods or operations may define separate constraints that limit their specific sets of acceptable values beyond those defined by property constraints.

If for a possible value no property modification value requirement is specified, the implementation may either accept or reject that value when provided as modification value.

The semantics of the modification operation or method may define how modification values are processed. Defining semantics includes the possibility that a modification value is only considered a hint, such that the value resulting from the instance modification differs from the provided modification value. If no specific semantics is defined, the default shall be that the modification value is carried over unmodified into the target instance.

7.13.2.11.4 Input parameter value requirement

Input parameter value requirements are input value requirements that may be specified for input parameters as part of method requirements in adaptation definitions. Value requirements shall not be specified for output parameters (for reasons detailed in 7.13.2.11.1).

Each input parameter value requirement shall be designated with a requirement level that determines the requirement for implementing the value as input parameter value.

An input parameter value requirement states that a specific value for an input parameter shall be implemented, that is, be accepted when provided as actual value in a method invocation.

Implementing an input parameter value does not preclude its rejection for reasons other than not being implemented, such as that the state of the managed environment does not currently allow the method execution request to be executed with the given set of input parameter values.

Input parameter value requirements shall only be specified for values that are within the value constraints established for the input parameter (see 7.13.2.10).

If for a particular parameter no parameter input value requirement is specified, the implementation behavior with respect to accepting input values for that parameter is undefined.

If for a possible value no input parameter value requirement is specified, the implementation behavior with respect to accepting that value as input is undefined.

7.13.3 Requirements for definitions of adaptations of ordinary classes and associations

7.13.3.1 General

Subclause 7.13.3 defines requirements for the definition of adaptations of ordinary classes and for the definition of adaptations of associations. These requirements apply in addition to the requirements defined in 7.13.2 for the definition of adaptations of all kinds of classes.

7.13.3.2 Definition of method requirements

7.13.3.2.1 General

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 that determines the requirement for implementing the method.

For the definition of requirements for parameters and method return values the requirements of 7.13.2.10 apply.

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.

Note that the Required qualifier for methods means that the method return values must not be Null; this does not imply a requirement to implement the method.

As part of a method requirement, a profile shall state requirements for all method parameters, each time repeating (from the schema definition of the adapted class) the effective values of the In and Out qualifiers and — if present — that of the Required qualifier.

NOTE This requirement aims at relieving profile consumers from analyzing the schema for respective requirements.

In addition, for each input parameter, input value requirements may be specified; for details, see 7.13.2.11.4.

Profiles should not replicate requirements from the schema or from base profiles unless needed for establishing additional requirements of the subject profile.

7.13.3.2.2 Requirements for the specification of constraints on methods and their parameters

The base set of permissible parameter and method return values is defined in the schema definition of the adapted class and/or its superclasses; as a matter of principle, schema definitions cannot be extended by profiles.

A profile may specify constraints and requirements for methods and their parameters (including method return values) as part of the method requirements.

Any such constraints and requirements shall apply in addition to, but shall not contradict, any constraints and requirements defined in the adapted class, its superclasses, and in base adaptations.

Different rules are established for the definition of such constraints for output parameters and method return values, as opposed to those for input parameters:

- For output parameters and method return values, profiles shall not specify method requirements that extend the set of permissible values as constrained in base adaptations, but may specify method requirements that further constrain that set. This rule ensures that the value set cannot be extended, and a client of a base adaptation never receives output values outside of the constraints established by base adaptations, even if an adaptation based on the base adaptation is actually implemented.
- For input parameters, profiles shall not specify method requirements that further constrain the set of permissible input values as constrained in base adaptations, but may specify method requirements that extend that set. This rule ensures that the permissible input value set cannot

be reduced, and conforming input values supplied by a client of a base adaptation are always to be accepted by the profile implementation, even if actually a derived adaptation is implemented.

However, note that this rule does not prohibit constraining the base set of permissible input values defined by the *schema definition* of the adapted class and/or its superclasses. In other words, a profile may specify method requirements constraining the base set of permissible input values for a property as established by the schema definition of the adapted class and/or its superclasses, such that only a smaller set of values is required to be accepted by a profile implementation. This applies likewise for property values of adaptation instances that are required as input value. Particularly, in adaptations modeling acceptable input parameter values, a profile may reduce the set of properties and their supported value ranges with respect to those defined by the adapted class and/or its superclasses, such that only the properties and value ranges established by the profile are required to be accepted by a profile implementation.

Profiles may specify the semantics of specific values of method input parameters (including values of properties in input instances) within the constraints already defined by the schema definition and base profiles. For example, for a method defined for the purpose of modifying an adaptation instance with an instance input parameter (that may or may not be an embedded instance), a profile may define that the value Null for properties in the input instance means not to change the value in the target instance.

NOTE This redefinition of the meaning of specific values is not generally possible for *instance modification operations* (see 7.13.3.3.4), because their semantics are established by the defining operations specification and usually require that all values from the input instance are to be carried over as given into the target instance. For that reason it might occasionally be advantageous to define methods with similar semantics as the creation and modification operations, but with more flexibility with respect to interpreting client provided input values, including the case to interpret values of certain input parameters as patterns or as suggestions, but not as strict value requirements.

In any case the schema definition of the adapted class, its superclasses, or any base adaptation may specify rules that establish limitations for the definition of such constraints in general, or under certain conditions.

NOTE These rules enforce polymorphic behavior of methods with respect to the method requirements defined in profiles. However, they do not enforce polymorphic behavior of methods with respect to the base set of permissible parameter value defined by the schema. This approach addresses the situation that schema definitions frequently define large value sets for input parameters with the intention that implementations constrain that value set to those values supportable by the implementation. Likewise, in the case where the input parameter is defined to be an (embedded) instance, that needs to be constrainable to instances of subclasses, to instances only containing values for a subset of the defined properties, and/or to instances where for specific properties the value set is constrained.

7.13.3.2.3 Management domain context of methods

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 the text shall be rephrased as standard English text.

In the schema, method semantics are typically only described with respect to the CIM model. The semantics described in the profile shall not contradict those defined in the schema. In addition — because profiles need to describe the relationship between the CIM model and the managed environment represented by that CIM model — in profiles it is generally not sufficient to describe only the expected state of the CIM model after the method execution completes. Instead, profiles should detail the required changes on managed objects in the managed environment that cause corresponding changes in the CIM instances that represent the managed objects.

For example, if an Example Fan profile requires that a fan is active as an effect of executing the RequestStateChange( ) method on the instance of the Fan adaptation representing the fan if the value of the RequestedState parameter is 2 (Enabled), that profile shall explicitly state as part of the required method semantics that the represented fan shall be activated, and not just that the value of the EnabledState property in the representing Fan instance shall be 2 (Enabled). The purpose of this requirement is to precisely instruct the implementer about the desired behavior in the managed environment, and not just about expected changes in the model representation of the managed environment. Of course, in addition the property requirements for the EnabledState property of the Fan adaptation need to separately state that the value shall be 2 (Enabled) if and only if the fan is active. For further rationale, see 6.6.3.

7.13.3.2.4 Specification of the reporting of method errors

The rules for the specification of reporting of operation errors defined in 7.13.3.3.6 shall be applied.

7.13.3.3 Definition of operation requirements

7.13.3.3.1 Operations specification

Profiles shall select DSP0223 as the operations specification, and define their operation requirements with respect to operations defined in DSP0223.

NOTE This requirement was introduced in version 1.1 of this guide in order to foster more protocol independence in profiles.

7.13.3.3.2 General

For each adaptation it defines, a profile shall define operation requirements. The operation requirements shall be stated with respect to the operations defined in DSP0223.

Each operation requirement shall be designated with a requirement level that determines the requirement for implementing the operation.

Profiles shall not define operation requirements for the operation(s) defined by the operations specification that request the execution of methods (such as the InvokeMethod( ) operation defined in DSP0223); instead, such operations are implicitly required if the profile defines any method requirements (see 7.13.3.2).

7.13.3.3.3 Specification of operation requirements for instance creation operations

The operations specifications (see 7.13.3.3.1) allow the creation of CIM instances based on input CIM instances provided by clients. In general, it is not required that values are provided in the input CIM instance for all properties; however, profiles may specify requirements for implementing specific initialization values (see 7.13.2.11.2).

As part of operation requirements for instance creation operations, profiles may specify

- preconditions that an input value is required to be provided in the input instance, or that an input value is not permitted to be provided in the input instance; such preconditions may be tied to other conditions specified by the profile.

  NOTE Operations specification define that provided values need to be reflected in the created instance, and how values of properties for which the input instance does not exhibit a value are to be determined for the created instance. For that reason the reinterpretation of specific values of input properties that is possible for input parameters of methods (see 7.13.3.2.2) is not admissible for operations.

- property value initialization constraints unless such are established by the schema (for example, by means such as the PropertyConstraint qualifier — see DSP0004).
- the effects of the operation with respect to the managed object to be created in (or to be added to) the managed environment.

NOTE An operations specification can specify semantics for the instance creation operations with respect to the resulting new instance.

- error reporting requirements as detailed in 7.13.3.3.6.

The specification of profile requirements for accepting input values for key properties in input instances for instance creation operations is not recommended, except for reference properties. An implementation is free to ignore any client provided value for a key property, except those for key reference properties. Clients should abstain from providing values for key properties other than reference properties in input instances for instance creation operations.

NOTE The reason behind this requirement is that the implementation is responsible for ensuring the uniqueness of instances. If clients were allowed to dictate key property values, clashes of instance creation requests from independent clients would be predestined.

For the creation of CIM instances it is of overriding importance that the lifecycle of a CIM instance is directly tied to the existence of a managed object in the managed environment that is represented by the CIM instance; see 6.6.2. A CIM instance can only be created if a respective managed object can be created (or added to the managed environment) such that the new CIM instance representing that managed object conforms with all values given by the input CIM instance with initialization constraints applied; for implementation requirements on instance creation operations, see 9.3.3.2.2.

7.13.3.3.4 Specification of operations requirements for instance modification operations

The operations specifications (see 7.13.3.3.1) allow modification of some or all property values of an instance. An operations specification also can specify semantics for the instance modification operations with respect to the resulting modified instance. Profiles may specify requirements for implementing specific modification values (see 7.13.2.11.3).

As part of operation requirements for instance modification operations, profiles may specify

- designations for specific properties to be either modifiable or non-modifiable.
  - Key properties are non-modifiable and shall not be designated as modifiable
  - Designations already specified in base adaptations should not be repeated or changed
  - Through such designations profiles may limit the effects of modification operations such that only the values of certain properties are affected.
- preconditions that an input value is required to be provided in the input instance, or that an input value is not permitted to be provided in the input instance; such preconditions may be tied to other conditions specified by the profile.

  NOTE Operations specification define that provided values need to be reflected in the created instance, and how values of properties for which the input instance does not exhibit a value are to be determined for the created instance. For that reason the reinterpretation of specific values of input properties that is possible for input parameters of methods (see 7.13.3.2.2) is not admissible for operations.
- the effect of property modifications with respect to the managed object to be modified in the managed environment unless these are apparent (for example by respective mappings of specific property values to respective states of the managed object)

  NOTE An operations specification can specify semantics for the instance modification operations with respect to the resulting modified target instance.
- error reporting requirements as detailed in 7.13.3.3.6.

For the modification of CIM instances it is of overriding importance that a CIM instance is the representation of (an aspect of) a managed object in the managed environment; see 6.6.2. A CIM instance can only be modified if the managed object represented by that CIM instance can be modified such that the CIM instance representing that modified managed object conforms with all values given by the input CIM instance; for implementation requirements on instance modification operations, see 9.3.3.2.3.

7.13.3.3.5 Specification of operation requirements for deprecated operations

Profiles shall not define operation requirements for operations that are marked as deprecated in the operations specification (see 7.13.3.3.1), except within revisions of existing profiles that retain an operation requirement for an operation that was marked as deprecated in the operations specification after the original version of the profile was released.

7.13.3.3.6 Specification of the reporting of operation errors

The operation requirements and method requirements specified by a profile should contain error reporting requirements.

Each error reporting requirement shall address a particular error situation.

Each error reporting requirement shall be designated with a requirement level that determines the requirement for implementing the error reporting requirement as part of implementing the method or operation.

Because in profiles error reporting requirements are a part of operation requirements or method requirements, each error reporting requirement specified in a profile shall be related to an error reporting requirement specified by the operations specification (see 7.13.3.3.1) as part of the definition of the operation. This also applies for method requirements if the method invocations are initiated through an operation; otherwise, error reporting requirements for methods shall be specified in context of an error reporting requirement established by the operations specification for method invocations.

The error situations addressed by error reporting requirements can overlap. For example, if an instance is not accessible, that may be caused by security reasons, by technical reasons or by other kinds of failures. Profiles may specify error reporting requirements with a relative order to each other, such that a particular error reporting requirement applies before other error reporting requirements. For example, in the case where an instance is not accessible for several reasons such as security reasons and several technical reasons, a profile could state that the error reporting requirement for reporting the security reason is to be applied before any other error reporting requirement.

Note that the operations specification may already have established a relative order among the error reporting requirements that it specifies. In this case, if the profile establishes a order among the profile specified error reporting requirements, that shall be in compliance with the order specified by the operations specification.

Profile should define each error reporting requirement through one or more standard messages, as follows:

- If the operations specification (see 7.13.3.3.1) defines error reporting requirements by means of standard messages, each error reporting requirement shall reference a standard error message (that is, a standard message defined in a DSP0228 conformant message registry with a type of "ERROR") required by the operations specification for the subject operation that addresses the error situation to be reported.
- If the operations specification (see 7.13.3.3.1) defines error reporting requirements by means of CIM status codes, each error reporting requirement shall reference a standard error message defined in DSP8016 that is compatible to a CIM status code required by the operations specification that is applicable in the error situation to be reported. A compatible standard error message shall exhibit — through the value of the CIMSTATUSCODE element — a CIM status code that applies in the error situation, and shall itself be applicable in the error situation to be reported.

- In cases where a mapping of CIM status codes to messages defined in DSP8016 is not possible, an error reporting requirements may directly reference the CIM status code instead of a standard error message.
- In addition, in all previous cases, an error reporting requirement may refer to one or more additional standard error messages that apply in the error situation to be reported. These messages are typically defined in a message registry that is separate from that used by the operations specification (see 7.13.3.3.1) and that contains definitions of messages that are more specific with respect to the domain addressed by the profile.
- Profiles may provide additional descriptions as part of error reporting requirements that detail the error situation in the context of which an error reporting requirement applies with respect to the management domain addressed by the profile. However, such additional descriptions are to be understood as implementation hints as to when — with respect to the management domain — an error reporting requirement applies. The additional descriptions shall not be understood as a constraint on the error situation that is described by the standard error messages and CIM status codes. Particularly, clients receiving an error indicator in the form of a set of standard error messages and a CIM status code shall only rely on the description provided directly through these elements. Clients shall not make assumptions based on the additional descriptions provided in profiles, other than that these describe single potentially possible error situations out of the typically much larger set described by the standard error messages and the CIM status code.

NOTE The implementation requirements resulting from error reporting requirements are detailed in 9.3.3.4.

7.13.3.3.7 Operation requirements related to associations

A profile shall define operation requirements for operations that enable association traversal as part of adaptations of classes that are referenced by association adaptations; typically such classes are ordinary classes.

The requirements for association traversal operations with respect to a particular association adaptation shall be specified separately as part of each referenced adaptation.

The requirements for association traversal operations of a particular adaptation of a class referenced by one or more association adaptations may be specified separately for each referencing association adaptation.

For example, consider a profile defines a System adaptation of the CIM_System class, a Device adaptation of the CIM_LogicalDevice class, and a SystemDevice adaptation of the CIM_SystemDevice association associating the System adaptation and the Device adaptation. If the association traversal operation requirements specified on the System adaptation with respect to the SystemDevice association may differ from those specified on the Device adaptation, they need to be separately specified.

Furthermore, if the profile had also defined a SystemPackaging adaptation of the CIM_SystemPackaging class, and if the association traversal operation requirements specified on the System adaptation targeting the Device adaptation through the SystemPackaging adaptation differ from those through the SystemDevice association adaptation, they need to be separately specified as well.

There is no implied requirement for an association adaptation to be implemented if one or more of the referenced adaptations are implemented. Similarly, the implementation of referenced adaptations is not implicitly required if an association adaptation is implemented. For that reason, profiles should ensure that all adaptations required to express a certain relationship are required as a whole; the preferred modeling approach in this case are features (see 7.15).

For example, extending the previously described situation with a mandatory System adaptation associated via a SystemDependency association adaptation to a Device adaptation, a profile should ensure that if the Device adaptation is implemented, then the SystemDevice adaptation is required to be implemented as well. For example, this could be achieved by defining the SystemDevice adaptation with the conditional exclusive requirement level, with the condition stating that the optional Device adaptation is implemented. Another more explicit approach could be defining an optional DevicesExposed feature, and define both the SystemDevice and the Device adaptations as conditional exclusive, with a feature implementation condition on the DevicesExposed feature.

7.13.3.3.8 Management domain context for operations

For write operations (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. Instead, profiles should detail the required changes on managed objects in the managed environment that cause corresponding changes in the CIM instances that represent the affected managed objects.

For example, if an Example Fan profile requires that a fan is active as an effect of executing the ModifyInstance( ) operation, that profile shall explicitly state as part of the required operation semantics that the identified fan shall be activated if the value of the EnabledState property in the input instance is 2 (Enabled), instead of repeating requirements from the operations specification (such as that the instance identified by the input instance shall adopt the values from the input instance) and/or the schema. The purpose of this requirement is to precisely instruct implementers about the desired behavior in the managed environment, and not just about expected changes in the model representation of the managed environment. Of course, the property requirements for the EnabledState property of the Fan adaptation need to separately state that the value shall be 2 (Enabled) if and only if the fan is active. For further rationale, see 6.6.3.

7.13.3.4 Definition of instance requirements

An instance requirement defines how (and in some cases also under which conditions) managed objects are to be represented by adaptation instances.

The definition of an adaptation in a profile models a particular managed object type or an aspect thereof; see 7.13.2.2. The implementation selects managed objects for representation. The definition of the adaptation implies the instance requirement to represent the selected managed objects as respective adaptation instances; profiles are not required to restate this implied instance requirement.

In addition, profiles may define the conditions in the managed environment that require the exposure of adaptation instances in namespaces; however, profiles should exercise care when stating such instance requirements in order to avoid requirements that cannot be satisfied.

For example, in the context of an Example Fan profile, consider an instance requirement phrased as follows: "Each fan shall be represented by a Fan instance." (where "fan" refers to fans in managed environments, and "Fan" refers to the Fan adaptation defined in that Example Fan profile). It is possible that some fans in the managed environment do not exhibit a management instrumentation that would enable a profile implementation to actually discover and control those fans. In these cases a profile implementation would not be able to comply with the specified instance requirement, because it can neither detect nor manage those fans without management instrumentation.

7.13.3.5 Metric requirements

Profiles may define metric requirements. Metric requirements shall be stated as part of class adaptations. These adaptations may be based on adaptations defined in the same profile, or in other profiles such as the *Base Metrics Profile* (see DSP1053).

The metric requirements shall be based on referenced metric definitions that are defined in metric registries. Besides formal requirements for the specification of metric definitions, DSP8020 also defines requirements for the implementation of metrics. These implementation requirements apply for profile implementations if a profile defines metric requirements by referencing metric definitions in metric registries that are compliant with DSP8020.

If necessary, as part of their metric requirements within adaptations profiles may amend the referenced metric definitions from metric registries. For example, such amendments may be necessary in order to refine the metric semantics and establish the context with the incorporating adaptation. In particular, this is required in the context of more generically defined metrics in metric registries. On the other hand, specific metric definitions in metric registries in many cases already define all necessary implementation requirements, such that referencing the registry-based definition along with the implementation requirements imposed by DSP8020 are sufficient for the purposes of the subject profile.

Profiles shall apply one of the following approaches for the definition of metric requirements:

- Managed object only (requires DSP1053, with either direct or indirect reference)

  With this approach, the metric requirements are defined as part of an adaptation that models the managed object type for which the metric applies, by

  – basing that adaptation on the MonitoredElement adaptation defined in the Base Metrics profile (see DSP1053), and

  – referencing in the same adaptation one or more metrics defined in a metric registry.

  This is the most compact approach because most of the metric related implementation requirements are implied from DSP1053. Specifically, the MonitoredElement adaptation from the Base Metrics profile implies implementation requirements for other adaptations defined in the Base Metrics profile, such as the BaseMetricDefinition adaptation, the BaseMetricValue adaptation, and their relationships. The adaptations from the Base Metrics profile also define how requirements from the metric definition in the metric registry apply in their context.

- Managed object and metric definition (requires DSP1053, with either direct or indirect reference)

  With this approach, the metric requirements are defined as part of a metric adaptation (an adaptation of the CIM_BaseMetricDefinition class or a subclass of that) by

  – basing that adaptation on the BaseMetricDefinition adaptation or on the AggregationMetricDefinition adaptation defined in the Base Metrics profile (see DSP1053),

  – referencing in the same adaptation one or more metric definitions defined in a metric registry (see DSP8020 for requirements on the specification of metric registries and their use), and

  – defining one or more adaptations based on the MonitoredElement adaptation defined in the Base Metrics profile modeling the entities for which the metrics apply, along with related association adaptations based on the MetricDefForME adaptation defined in the Base Metric profile that relate the managed elements with their metric definitions.

  This is a less compact, but more flexible, approach. In addition to its own requirements, the BaseMetricDefinition adaptation from the Base Metrics profile implies additional implementation requirements for related adaptations defined in the Base Metrics profile, such as the BaseMetricValue adaptation and its relationships. However, with this approach the subject profile is required to establish the context to one or more managed elements through its adaptations based of the MetricDefForME adaptation. Again, the adaptations from the Base Metrics profile also define how requirements from the metric definition in the metric registry apply in their context.

- Complete approach (DSP1053 not required, but possible)

  With this approach, the subject profile defines all aspects of the metric requirements through one or more adaptations, and with or without referencing other profiles. At least one the metric related adaptations is required to be based on a metric definition in a metric registry, and establish the usage context of that registry-based metric definition for the modeled managed object types.

This is the most flexible approach. It does not require referencing DSP1053, but requires the most extensive definitions in the subject profile. The subject profile may or may not define its metric-related adaptations based on adaptations defined in DSP1053 or in other profiles. If so, then the requirements of the base adaptations are imposed as usual. If not, then the subject profile itself must define all metric-related requirements such as interpretation rules or value constraints of certain metric-related properties, or as relationships between metric-related adaptations.

7.13.3.6 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 in establishing such requirements, because they might reduce the set of managed environments for which the profile can be implemented.

7.13.3.7 ACID requirements

Profile authors should be aware that protocols, WBEM server 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 server. Profile-defined ACID requirements shall be stated in a protocol-agnostic manner.

NOTE ACID properties for operations and methods are defined in operations specifications (see 7.13.3.3.1).

If profiles define ACID requirements, these shall not contradict other specification rules established by this guide, such as requirements for the specification of instance requirements (see 7.13.3.4) or that for the specification of operations requirements (see 7.13.3.3).

7.13.4 Requirements for the definition of indication adaptations

7.13.4.1 General

The requirements defined this subclause apply in addition to the requirements defined in 7.13.2 for the definition of adaptations of all kinds of classes.

The approach detailed in this subclause aims at relieving profiles that define indications from having to define many of the infrastructure elements related to indications, such as indication filters and filter collections. This is because such infrastructure elements are already implied by definitions of DSP1054. Particularly in the case of alert indications, the specification effort in profiles is typically reduced to just define an adaptation based on the AlertIndication adaptation defined DSP1054, along with a reference to an alert message for each event that is to be reported.

A profile that defines indications may reference DSP1054; if a profile references DSP1054, it shall comply with the requirements defined in DSP1054 for referencing profiles. A profile referencing DSP1054 may define its indication adaptations based on those defined in DSP1054. As usual, the "based on" relationship to basic indication adaptations defined in DSP1054 may be indirect, with intermediate other base adaptations. In either case, the requirements of the base indication adaptation defined in DSP1054 implicitly applies, including the requirements for related indication filters and filter collections.

An alert indication adaptation that is defined based on the AlertIndication adaptation defined in DSP1054 may reference alert messages defined in a message registry. For each message reference, the alert indication adaptation shall state the message registry reference (see 7.12) referring to the defining message registry, and uniquely identify the message by stating its message id. The message id is the concatenation of the value of the PREFIX attribute and the SEQUENCE_NUMBER attribute from the

MESSAGE_ID element that defines the alert message within the message registry. Furthermore, the alert indication adaptation shall specify how the definitions of the referenced alert messages apply, unless such information is already sufficiently provided by the definition of the AlertIndication adaptation defined in DSP1054, by the respective alert message definitions, by the Message Registry XML Schema Specification (see DSP8020), or by a combination of these definitions. For rules on how to conform with these requirements in profile specification documents, see 10.4.7.4.3.

7.13.4.2 Indication-generation requirements

For each indication adaptation one or more indication-generation requirements shall be defined. Each indication-generation requirement shall express the situation that causes the indication to be generated; in most situations such descriptions just refer the event reported by the indication, but additional constraints may apply.

The basic indication adaptations defined in DSP1054 already define indication-generation requirements. As with any requirement defined by a base adaptation, the indication-generation requirements defined by base indication adaptations (such as those defined in DSP1054) implicitly apply in context derived indication adaptations; however, if needed, a derived indication adaptation may refine the indication-generation requirements of its base indication adaptation(s).

7.13.5 Abstract class adaptation

Abstract class adaptations are class adaptations with an implementation type of "abstract". Any class that is not an abstract class adaptation is termed a concrete class adaptation.

One purpose of abstract class adaptations is to serve as a common endpoint for generic association adaptations, such that the relationship applies to any class adaptation based on the abstract class adaptation and the definition of specific association adaptations for every possible endpoint can be avoided.

Another purpose of abstract class adaptations is grouping the common requirements of other class adaptations. Instead of repeating the common requirements in each specific class adaptation the common requirements are specified in an abstract class adaptation, and each specific class adaptation is based on that abstract class adaptation.

Abstract class adaptations are not directly implemented; instead, their requirements are propagated into class adaptations that are based on them. For details, see clause 9.

Each class adaptation adapting an abstract class from a schema shall be designated as an abstract class adaptation, with one exception:

> A profile may define a concrete (non-abstract) adaptation of an abstract class, if in addition it states a concrete class derived from the adapted class that shall be implemented if the profile implementation does not need a more specific derived class. For example, a profile may define an XxxComponent adaptation of the (abstract) CIM_Component class and state that the CIM_ConcreteComponent class shall be implemented if the implementation does not require a more specific association derived from CIM_Component. This specification approach enables implementations to define their own implementation classes derived directly from the abstract CIM_Component association (instead of being forced to base their implementation class on the concrete CIM_ConcreteComponent association).

7.13.6 Trivial class adaptation

A trivial class adaptation does not define additional requirements beyond those defined by its 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 adaptations based on them. Another typical use of a trivial class adaptation is introducing a concrete equivalent of an abstract class adaptation in the case where no additional requirements need to be defined beyond those defined by the abstract class adaptation.

7.13.7 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. In this case a conformant implementation of that profile's adaptation of the CIM_ComputerSystem class is only required to show non-Null values for the properties exposed by the CIM_ComputerSystem class that are either key properties, or that are properties with the REQUIRED qualifier having a value of True.

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 defining requirements and constraints only for a subset of the properties exposed by 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 such as 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 server.

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 profile 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 be based on that of the referenced generic CPU profile

  Typically this implies a separate profile implementation of the referenced generic CPU profile, exploiting management instrumentation provided by the virtualization platform in the context of which virtual processors exist. The set of instances provided by that profile 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 virtual processors now are visible through the interface defined by the generic CPU profile; consequently, a client could manage the virtual processors through that interface in the same way as in the physical case. However, it should be noted that in this case the set of CIM_Processor instances is disjoint from that representing the host processors in the physical case.

As detailed in clause 9, a profile implementation is required to conform to the definitions of the profile and those of referenced profiles. More specifically, an implementation of an adaptation is required to satisfy all requirements of all base adaptations, including instance requirements.

7.14 Requirements for profile registration

The CIM schema defines classes that enable the representation of implemented profile versions and their relationships, such as the CIM_RegisteredProfile class and the CIM_ElementConformsToProfile and CIM_ReferencedProfile associations. The Profile Registration profile (see DSP1033) defines a model for the representation of implemented profile versions and their relationships by defining the use of these classes; see DSP1033 for details.

Concrete profiles except the Profile Registration profile (see DSP1033) shall reference the Profile Registration profile (see DSP1033) as a mandatory profile.

This implies that the central class adaptation (see 7.9.3.2) conforms to the requirements for central classes defined by the Profile Registration profile (see DSP1033), that the scoping class adaptation (see 7.9.3.3) conforms to the requirements for scoping classes defined by the Profile Registration profile (see DSP1033), and that the adaptation of the CIM_RegisteredProfile class modeling the profile registration of the subject profile conforms with the requirements of the CIM_RegisteredProfile "profile class" defined by the Profile Registration profile (see DSP1033).

NOTE 1 The requirements for central classes and scoping classes defined by the Profile Registration profile (see DSP1033) imply the implementation of a profile advertisement methodology.

NOTE 2 It is expected that a future version of the Profile Registration profile (see DSP1033) is defined based on version 1.1 (or later) of this guide, and defines adaptations such as a CentralElement, a ScopingElement and a ProfileRegistration adaptation that could serve as base adaptations for the central class adaptation, the scoping class adaptation and the profile registration adaptation of referencing profiles. This will allow defining the requirements related to profile registration and to central class adaptations and scoping class adaptations more precisely.

Abstract profiles may reference DSP1033 as a mandatory profile; if so, the requirements of DSP1033 apply for the (implicit) profile implementation of the abstract profile as part of a concrete profile derived from the abstract profile, as well as for the profile implementation of the concrete profile itself because that is also required to reference DSP1033 as a mandatory profile.

NOTE 1 This enables clients to be written against an abstract profile without requiring knowledge about the implemented concrete profile derived from the abstract profile.

NOTE 2 Version 1.0 of this guide was unclear about whether or not abstract profiles were allowed to refer to DSP1033.

In any case, the requirements of 7.9.3.2, 7.9.3.3 and 7.9.3.4 apply.

7.15 Requirements for the definition of features

7.15.1 Introduction

A feature is a named profile element; the rules defined in 7.2.2 apply. A feature 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 element conditional on the implementation of the feature.

7.15.2 General feature requirements

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

Profiles 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.

7.15.3 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 7.2.2.

7.15.4 Feature requirement level

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

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.

7.15.5 Feature granularity

Feature granularity affects the discoverability 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 within a particular profile implementation. Feature discoverability is defined at a global level, such that if the feature is available, it is available for all instances affected by definitions that depend in the feature.
- Features with instance granularity are available only for certain instances. Feature discoverability is defined at an adaptation instance level, such that the availability of the feature is indicated only for certain adaptation instances that conform to additional requirements.

Profiles shall define the granularity of each feature by indicating whether the feature is defined with either profile granularity or with instance granularity; if defined with instance granularity, profile shall state an adaptation and the conditions for which instances of that adaptation the feature is required to be available.

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 profile's Fan adaptation.

In another example (detailed in 7.15.1), 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.

7.15.6 Feature discovery

Feature discovery aims at enabling clients to discover the availability of features.

It is highly recommended that a profile defines at least one mechanism that facilitates discovery of a feature availability as part of a profile implementation.

Each discovery mechanism shall be defined such that the availability and the unavailability of the feature can be discovered.

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

An example of a feature discovery mechanism is a specific value constraint for a property value in a capabilities instance. For example, an Example Fan profile could define the preferred discovery path for the availability of its FanElementNameEdit feature by requiring that if the FanElementNameEdit feature is available for a fan then there is an associated instance of the CIM_EnabledLogicalElementCapabilities class for which the value of the ElementNameEdit property is True. These capabilities instances could be combined into one shared instance that is associated to those Fan instances for which the feature is available.

The discovery mechanism described 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 the context of an instance for which the feature can apply. For example, this is the case for the Fan instances described in the last example in 7.15.5, but only in the case where the FanSpeedSensor feature is supported for those fans that are represented by Fan instances with an associated FanSpeedSensor instance.

7.15.7 Feature requirements

Feature requirements are the implementation requirements resulting from the commitment to implement a feature. The commitment can result from a deliberate decision of the implementer, but in the case of conditional features can also be the result of a True condition. Feature requirements are not defined as an integral part of the feature. Instead, they are specified as conditional requirements for other profile definitions such as referenced profiles, adaptations, property requirements, method requirements, operation requirements, or metric requirements. This approach enables the specification of profile elements that depend on more than one feature.

A profile shall define feature requirements in terms of requiring otherwise optional profile elements as conditional or conditional exclusive with feature implementation conditions (see 7.4.3), or by defining additional constraints. Profiles shall use the following mechanisms to define feature requirements:

- Defining profile elements as conditional or conditional exclusive with respect to the feature implementation; this applies to
  - profile references
  - otherwise optional, conditional or conditional exclusive profile elements within referenced profiles, such as features, adaptations, property requirements, or method requirements
  - adaptations
  - base adaptations
  - property requirements in adaptations
  - method requirements in adaptations
  - operation requirements in adaptations
  - error reporting requirements in adaptations
  - metric requirements in adaptations

- Defining constraints that depend on implementation of the feature

NOTE Clause 9 defines requirements for implementations of profiles, including those of conditional profile elements. See clause 9 for the implementation requirements resulting from features.

7.15.8 Feature example

Figure 8 shows two DMTF collaboration structure diagrams that detail the collaboration defined by an Example Fan profile. For respective diagrams of the Example Profile Registration profile (referenced in both parts of Figure 8) and an Example Sensors profile (referenced in the lower part of Figure 8), see 7.13.2.1. For details on DMTF collaboration structure diagrams, see 8.3.4.

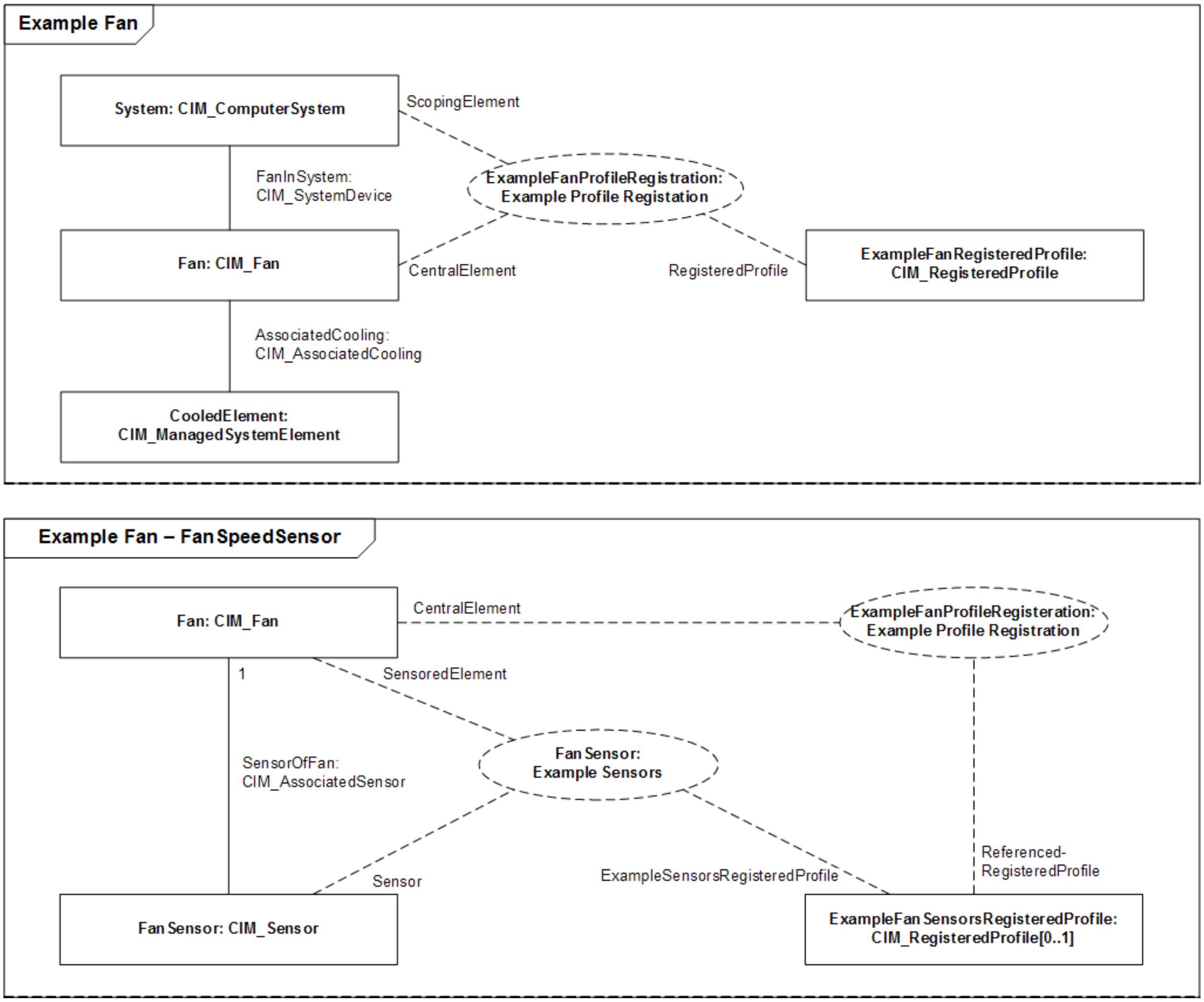


Figure 8 – Examples of DMTF collaboration structure diagrams

The upper diagram in Figure 8 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 Example 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 8 indicates that each fan sensor provides sensor information for exactly one fan.

The lower diagram in Figure 8 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 ExampleFanSensorsRegisteredProfile 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 fan speed sensors in the managed environment; this is an example of a managed environment condition (see 7.4.7). Consequently an implementer who implements the Example Fan profile for a particular type of managed environment (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 (for example, 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 which the profile is implemented. If that is the case, then the feature driven implementation requirements become effective and need to be implemented.

Furthermore, in this example it is assumed that individual fans in the managed environment may or may not have sensors. However, this cannot be expressed in the CSD, and in any case needs to be stated in the form of normative definitions in the Example Fan profile. A further assumption in this example 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 through a SensorOfFan association adaptation instance. The instance granularity of the feature in effect requires the profile implementation to provide feature-required elements only for those Fan instances that represent a fan with a sensor.

NOTE Features with instance granularity allow mandating presence of the feature only for the CIM representation of specific managed objects that exhibit a certain behavior or functional element (such as fans with sensors). 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.

A client would discover the presence of the FanSpeedSensor feature for a particular Fan instance by traversing from the Fan instance through SensorOfFan to FanSensor instances; the presence of such instances would indicate the presence of the FanSpeedSensor feature for the Fan instance.

An alternate discovery path for the FanSpeedSensor feature could be defined through the ExampleFanSensorsRegisteredProfile instance associated through the CIM_ReferencedProfile association to the ExampleFanRegisteredProfile instance representing the implemented version of the Example Fan profile. This is depicted in the lower part of Figure 8 on the right side by showing the ExampleSensorsRegisteredProfile adaptation of the Example Fan profile based on the ReferencedRegisteredProfile adaptation of the Example Profile Registration profile. The ReferencedRegisteredProfile adaptation in turn requires the implementation of the CIM_ReferencedProfile association to the CentralElement adaptation. Thus, a client inspecting an implemented version of the Example Fan profile as represented by a ExampleFanRegisteredProfile instance can detect that the FanSpeedSensor feature is implemented by traversing the CIM_ReferencedProfile association to a ExampleFanSensorsRegisteredProfile instance. If that instance exists, this indicates that the FanSpeedSensor feature is implemented in general; however, because in this example the FanSpeedSensor feature is defined with a granularity of "Fan instance", the feature is available only for those Fan instances that represent fans with sensors.

If the FanSpeedSensor feature is implemented, then all other profile definitions that are conditional on this feature effectively become implementation-required; see clause 9 for an algorithm allowing the determination of all implementation-required profile elements in the context of the profile implementation of one or more referenced 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 SensoredElement adaptation of the Example Sensors profile.

7.16 Requirements for the definition of use cases

7.16.1 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 perform a management task by means of the interface defined by the profile, and the effects on managed objects in a managed environment and their CIM representation in the course of performing that task.

A use case is a named profile element; the rules defined in 7.2.2 apply.

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 such as CIM clients or other external entities such as 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 parts of the profile such as the definition of adaptations. However, 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 applied by 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.

7.16.2 Requirements for the definition of state descriptions

State descriptions may be provided as part of a use case, but may be provided separately and be referenced other parts of the profile, particularly use cases.

State descriptions defined outside of a use case are named profile elements that describe the state of an instance of (a subset of) the model defined by a profile at a particular point in time.

State descriptions within a use case may be named for the purpose of referencing them within a across use cases defined in the same profile.

State descriptions should be stated in terms of adaptation instances, their properties with actual values, and by stating which managed object is represented. Only adaptation instances that are involved in the processing of referencing use cases need to be described. Likewise, for each stated adaptation instance the set of stated property value pairs may be constricted to those relevant in referencing use cases.

Within state descriptions, adaptation instances may be named for the purpose of referencing them. For a particular adaptation instance, these names are required to be unique only within the scope of the state description; in other words, the use of the same name for an adaptation instance in two unrelated state descriptions does not imply the same adaptation instance. References to adaptation instances should ensure that the context to their state description is established.

State descriptions may be expressed in the form of DMTF object diagrams; for details, see 8.3.7.

7.16.3 Requirements for the definition of preconditions

For each use case the preconditions shall be defined.

Preconditions are state descriptions (see 7.16.2) that describe the *initial* state of an instance of (a subset of) the CIM model defined by the profile.

Additional preconditions may be stated in terms of managed objects. In exceptional cases, preconditions may be stated exclusively in terms of the managed objects.

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

7.16.4 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 the subject profile or by referenced profiles in the form of method requirements.

For each use case step, the following types of provisions 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 input parameters relevant to the use case
- the expected effect on the managed environment
- the corresponding changes on the CIM model
- the names and values of output parameters relevant to the use case
- the expected return values, 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.

7.16.5 Requirements for the definition of postconditions

For each use case the postconditions should be defined if the execution of the use case caused changes in the CIM model defined by the profile.

Postconditions are state descriptions (see 7.16.2) that describe the *resulting* state of (a subset 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.

Additional postconditions may be stated in terms of managed objects. In exceptional cases, postconditions may be stated exclusively in terms of managed objects.

NOTE Note that as described in 6.6.3 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.

7.17 Backward compatibility

This subclause defines rules for maintaining backward compatibility between versions of profiles. 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 the context of a profile 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 shall not be introduced in new minor versions or in new update versions, except for error corrections. If incompatibilities are introduced in new minor versions or in new update versions as part of error corrections, each incompatibility shall be described from a client perspective, and shall state both the version it breaks, and the version introducing the incompatibility.

7.18 Definition of experimental content

A profile may designate definitions as experimental. In this case the rules about experimental content as defined in the "Document conventions" of this guide for experimental material shall be applied.

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

7.19 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.

For deprecated profile definitions the rules about deprecated content as defined in the "Document conventions" of this guide for deprecated material shall be applied.

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. However, minor revisions of profiles that use schema elements that are deprecated in a newer version of the schema are not obliged to be upgraded to the new schema version just for the purpose of changing to the replacement of the deprecated element.

8 Profile general conventions and guidelines

8.1 General

Clause 8 defines general conventions and guidelines that apply for all kinds of profiles, including those specified in form of profile specifications (as detailed in clause 9), or in the form of machine readable profiles. In any case with respect to the profile content the requirements detailed in clause 7 apply.

8.2 Linguistic and notational conventions

This subclause defines linguistic and notational conventions for textual definitions in profiles.

All words should be in lower case 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 Ethernet.
- The word is an acronym, such as CPU.
- The words are part of a profile name (see 7.6.2), such as Profile Registration.
- The word is a schema element, such as CIM_SystemDevice.

Phrases should not be concatenated into one word unless one of the following conditions is met:

- The word is the name of a named profile element (see 7.2.2), such as FanStateManagement or FanCapabilities.
- The word is a schema element, such as CIM_SystemDevice, EnabledState, or RequestStateChange( ).
- The word is an object name, such as MAINCPUFAN.

Elements of the managed environment and elements of the CIM model defined by the profile should be clearly distinguished. 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 to:

- 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.
- Object names should be all uppercase, such as in MAINCPUFAN.

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. Furthermore, assume an example situation where a Fan instance named MAINCPUFAN represents the fan of the main CPU within an example system.

Table 2 juxtaposes examples of recommended phrasing with examples of phrasing that is wrong or confusing.

Table 2 – Specification recommendations

Recommended	Not recommended (wrong, unclear or confusing)
"The Fan instance MAINCPUFAN represents the CPU fan." NOTE 1 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. NOTE 2 Fan identifies the Fan adaptation, MAINCPUFAN identifies a particular instance, and CPU fan identifies a managed object. Names of named profile elements (such as adaptations) are capitalized (see 7.2.2), object names should be all uppercase, and all other words are not capitalized unless required by normal English language.	"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 MAINCPUFAN is 2 (Enabled)." Alternative: "The EnabledState value in MAINCPUFAN is 2 (Enabled)."	"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)." Problem: The state of the managed object (the CPU fan) is being confused 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: The state of the managed object is being confused with the textual representation of a property value in the instance representing the managed object.
	"EnabledState shall match 2." Problem: The property name and the property value are not distinguished.

8.3 Conventions and guidelines for diagrams

8.3.1 General

Five types of diagrams are commonly used in profiles:

- EXPERIMENTAL: **DMTF collaboration structure diagrams** (see 8.3.4) show the structure of a profile or subset thereof, and the collaborations that this structure makes possible.
- EXPERIMENTAL: **DMTF adaptation diagrams** (see 8.3.5) show the adaptations defined by a profile or subset thereof, and possibly adaptations defined in referenced profiles.
- **DMTF class diagrams** (see 8.3.6) show the classes adapted by a profile (and possibly classes adapted by referenced profiles).
- DEPRECATED: **DMTF profile class diagrams** (see 10.3.3.2) show "profile classes" (see deprecation notice in 7.13.1). DMTF profile class diagrams are only admissible in revisions of existing profile specifications that maintain the traditional profile specification structure (see 10.3.3).

- **DMTF object diagrams** (see 8.3.7, also referred to as instance diagrams) show a set of related objects (or, more precisely, adaptation instances) at a point in time. Object diagrams may be associated with use cases, by showing how the use case affects properties and object relationships.
- **DMTF sequence diagrams** (see 8.3.8) show the interaction between adaptation instances in terms of methods and operations.

8.3.2 General diagram guidelines

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

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

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

8.3.3 Diagram color conventions

The color conventions as defined in this subclause should be applied for DMTF adaptation diagrams (see 8.3.5), DMTF class diagrams (see 8.3.6), DMTF profile class diagrams (DEPRECATED, see 10.3.3.2), and DMTF object diagrams (see 8.3.7). Deviations from the color conventions are permitted, but they shall be documented and consistently applied.

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

In DMTF adaptation diagrams this symbol may also be used to represent the "based on" relationship between adaptations. In DMTF object diagrams, inheritance relationships shall not be shown.

DEPRECATED

- 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. However, with CIM associations, an association endpoint is owned by the association itself; consequently, the former convention of showing a dot is incorrect, and is replaced by the conventions for aggregation and composition associations not showing the dot.

- 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. Because 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

EXPERIMENTAL

8.3.4 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 features), 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 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 = RegisteredProfileName [ WS "-" WS SubpartName WS
  SubpartType ]
  ```

  `RegisteredProfileName` shall be the registered name of the profile. `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 a feature, pattern, or scenario.
- Adaptations of ordinary classes or indication classes shall be represented as UML parts.

  It is not required that all adaptations defined by a profile are shown; instead, the selection of adaptations for display in one or more CSD diagrams is left to the profile author. Also, multiple CSD diagrams may be shown, each reflecting a sub-collaboration defined in the profile.

  Each UML part shall be shown as a solid rectangle (box), and shall be named as follows:

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

`AdaptationName` shall be the name of the ordinary class or indication adaptation, `ClassName` shall be the name of the adapted ordinary or indication class, and `PartMultiplicity` shall be the multiplicity of the part.

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 = AssociationAdaptationName *WSP ":" *WSP
AssociationClassName
```

  `AssociationAdaptationName` shall be the name of the association adaptation, and `AssociationClassName` shall be the name of the adapted association class.

  – If represented in a CSD, references defined by association adaptations shall be represented as UML endpoint names. UML endpoint names shall be shown as text at the ends of a UML connector.

  – If represented in a CSD, reference multiplicities shall be represented by UML endpoint multiplicities. The representation of reference multiplicities is required 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 ] *WSP ":" *WSP
ProfileName
```

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

  `ProfileName` shall be the name of the referenced 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.

- If represented in a CSD, 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.

  A UML role binding shall be shown as a dashed line connecting a UML collaboration use representing the profile that defines a base adaptation, and the UML part representing a class adaptation defined in the subject profile. A UML role binding shall be labeled close to the class adaptation end, as follows:

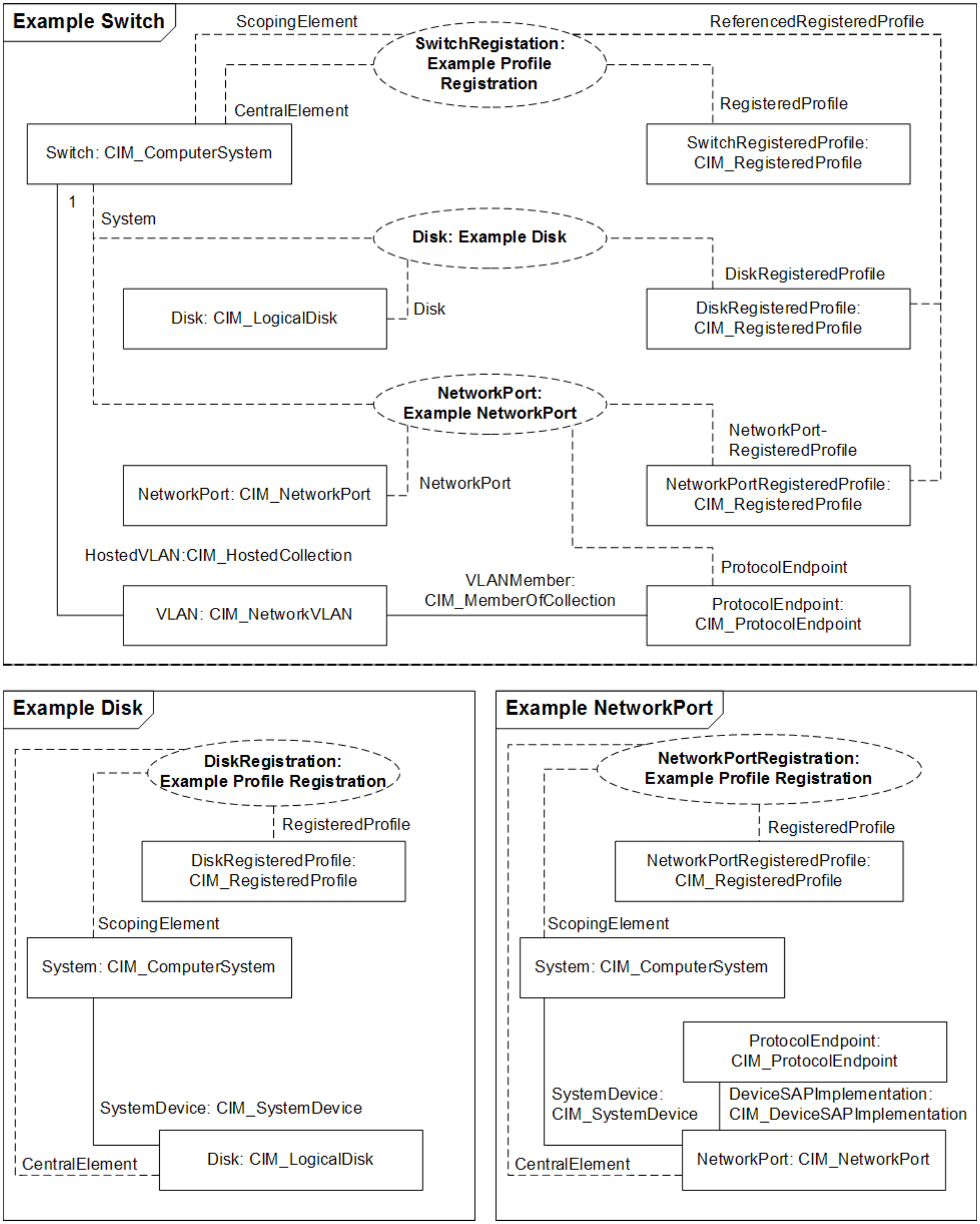
```
EndRoleName = BaseAdaptationName
```

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

  For a particular adaptation it is not required that any relationships to profiles defining base adaptations is shown through UML role bindings; the selection is left to the profile author.

- As an alternative to the use of UML collaboration uses and UML role bindings, the inheritance arrow may be used to show the relationship between an adaptation and its base adaptation(s).

3335 Figure 8 shows examples of three DMTF collaboration structure diagrams depicting collaborations
3336 defined by one autonomous profile and two component profiles.


Example Switch
ScopingElement
ReferencedRegisteredProfile
SwitchRegistation: Example Profile Registration
CentralElement
RegisteredProfile
Switch: CIM_ComputerSystem
SwitchRegisteredProfile: CIM_RegisteredProfile
1
System
Disk: Example Disk
DiskRegisteredProfile
Disk: CIM_LogicalDisk
Disk
DiskRegisteredProfile: CIM_RegisteredProfile
NetworkPort: Example NetworkPort
NetworkPort-RegisteredProfile
NetworkPort: CIM_NetworkPort
NetworkPort
NetworkPortRegisteredProfile: CIM_RegisteredProfile
HostedVLAN:CIM_HostedCollection
ProtocolEndpoint
VLANMember: CIM_MemberOfCollection
VLAN: CIM_NetworkVLAN
ProtocolEndpoint: CIM_ProtocolEndpoint
Example Disk
DiskRegistration: Example Profile Registration
RegisteredProfile
DiskRegisteredProfile: CIM_RegisteredProfile
ScopingElement
System: CIM_ComputerSystem
SystemDevice: CIM_SystemDevice
CentralElement
Disk: CIM_LogicalDisk
Example NetworkPort
NetworkPortRegistration: Example Profile Registration
RegisteredProfile
NetworkPortRegisteredProfile: CIM_RegisteredProfile
ScopingElement
System: CIM_ComputerSystem
ProtocolEndpoint: CIM_ProtocolEndpoint
SystemDevice: CIM_SystemDevice
DeviceSAPImplementation: CIM_DeviceSAPImplementation
CentralElement
NetworkPort: CIM_NetworkPort

3337

3338 **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 Example Profile Registration profile is referenced by the Example Switch profile. This is depicted by the UML collaboration use (dashed oval) named "`SwitchRegistration: Example Profile Registration`".
- The System adaptation is based on the CentralElement adaptation of the Example 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: Example 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 Example 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 `ReferencedRegisteredProfile` connecting the UML collaboration named "`SwitchRegistration: Example Profile Registration`" with the UML part (solid rectangle) named "`DiskRegisteredProfile: CIM_RegisteredProfile`". The latter corresponds to the DiskRegisteredProfile adaptation of the Example Disk profile, as depicted by the UML role binding named `DiskRegisteredProfile` 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.

EXPERIMENTAL

EXPERIMENTAL

8.3.5 DMTF adaptation diagram guidelines

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

The diagram color conventions defined in 8.3.3 apply.

For DMTF adaptation diagrams the following additional rules and conventions apply:

- DMTF adaptation diagrams shall show class adaptations (adaptations of ordinary classes, association classes, and indication classes).
- A DMTF adaptation diagram shall be labeled as follows:

  ```
  DADLabel = RegisteredProfileName [ WS " - " WS SubsetName ]
  ```

  RegisteredProfileName shall be the registered name of the profile. SubsetName may be used if the DMTF adaptation diagram shows a subset of adaptations defined by the profile; in this case, SubsetName should paraphrase the purpose of the shown subset of adaptations.

- If represented in a DMTF adaptation diagram, adaptations of ordinary classes or indication classes shall be represented as UML classes where the UML class name shall be the adaptation name. The following format shall be applied:

  ```
  BoxLabel = AdaptationName
            [ "(" *WSP "from" WS RegisteredProfileName *WSP ")" ]
            [ "{" *WSP "adapts" WS ClassName *WSP "}" ]
  ```

  AdaptationName shall be the name of the adaptation. If the adaptation is defined in a profile other than the subject profile, the "from" part shall be used and the referencing profile's registered profile name shall be stated as RegisteredProfileName. Unless the name of the adapted class is identical to the adaptation name prefixed with CIM_, the "adapts" part should be used and ClassName shall be the name of the adapted class.

- If represented in a DMTF adaptation diagram, adaptations of associations shall be represented as UML associations, or more specifically as UML aggregations or UML compositions if respective semantics apply from the schema definition of the adapted association. The UML association name shall be the name of the association adaptation. The following format shall be applied:

  ```
  AssociationLabel = AssociationAdaptationName
            [ "(" *WSP "from" WS RegisteredProfileName *WSP ")" ]
            [ "{" *WSP "adapts" WS AssociationClassName *WSP "}" ]
  ```

  AssociationAdaptationName shall be the name of the association adaptation. If the association adaptation is defined in a profile other than the subject profile, the "from" part shall be used and the referencing profile's registered profile name shall be stated as RegisteredProfileName. Unless the name of the adapted association class is identical to the adaptation name prefixed with CIM_, the "adapts" part should be used and AssociationClassName shall be the name of the adapted association class.

  – Reference properties required by association adaptations may be represented as UML association ends. If used, UML association ends may be shown as text at the ends of the UML association representing the association adaptation.

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

- In general, any adaptation defined by a profile should be depicted at most once in a DMTF adaptation diagram. The desire for depicting a particular adaptation more than once should be taken as an indicator that the definition of a separate adaptation is appropriate.
- DMTF adaptation diagrams should not show properties and methods.

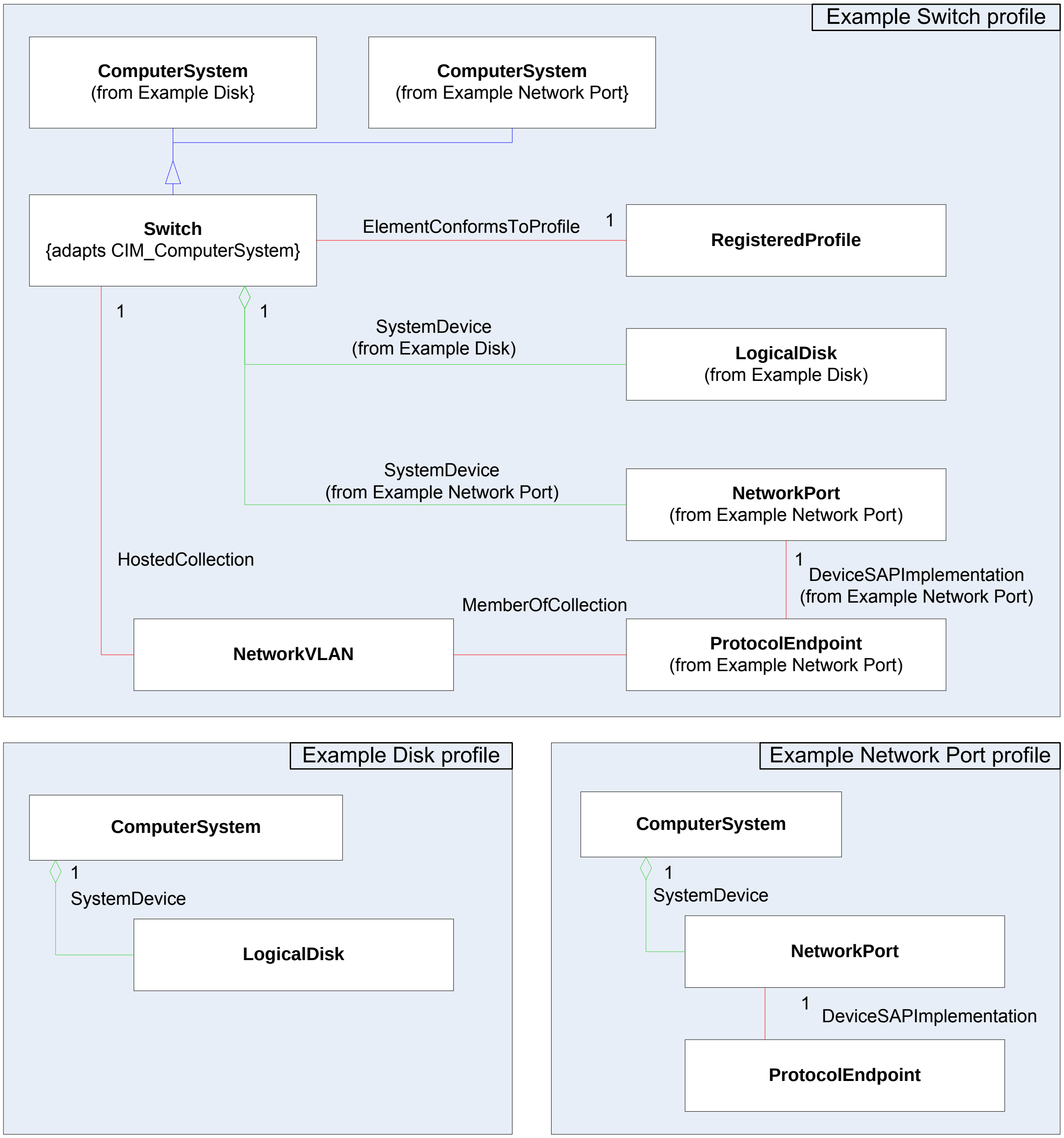


Figure 10 – Examples of DMTF adaptation diagrams

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

NOTE The shaded rectangles are not part of the conventions for DMTF adaptation diagrams as defined in 8.3.5; they are shown here such that multiple DMTF adaptation diagrams can be condensed into one diagram.

The upper part of Figure 10 shows the DMTF adaptation 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 adapts the CIM_ComputerSystem class, and is based on both the ComputerSystem adaptations defined in the Example Disk profile and in the Example Network Port profile.

EXPERIMENTAL

8.3.6 DMTF class diagram guidelines

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

The diagram color conventions defined in 8.3.3 apply.

DMTF class diagrams shall show adapted ordinary classes, adapted association classes and adapted indication classes.

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.

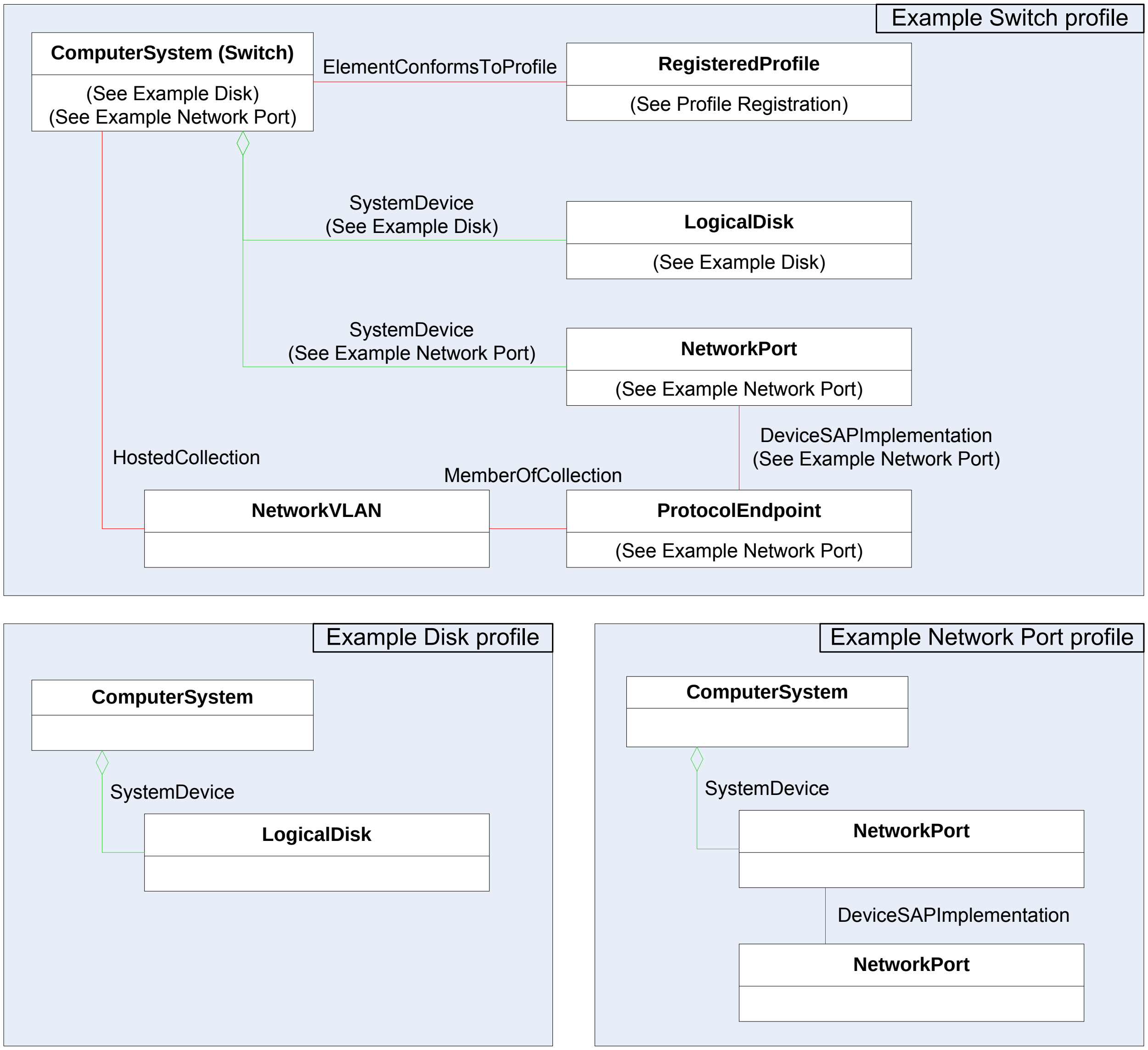


Figure 11 – Examples of DMTF class diagrams

Figure 11 shows examples of class diagrams from one autonomous profile and two component profiles.

NOTE The shaded rectangles are not part of the conventions for DMTF class diagrams as defined in 8.3.6; they are shown here such that multiple DMTF class diagrams can be condensed into one diagram.

The upper part of Figure 11 shows the 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.

8.3.7 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 *WSP ] "/" *WSP AdaptationName /
 ":" *WSP ClassName /
 "/" *WSP AdaptationName ":" *WSP ClassName

InstanceName = *[("A"-"Z")/("0"-"9")/"_"]
```

The `AdaptationName` 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 uppercase 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. `ClassName` may be used in addition to `AdaptationName`; it may also be used instead of the `ClassName` when presenting the use of a class for which an adaptation is not required by the subject profile.

Examples:

```
SYSTEM1 / System                    ; InstanceName/AdaptationName
SYS_2: CIM_ComputerSystem           ; InstanceName:ClassName
CLUSTER/Cluster: CIM_AdminDomain        ; all three components
/VirtualSystem                      ; /AdaptationName
: CIM_ComputerSystem                ; :ClassName
```

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 *WSP PropertyAssignment *WSP PropertyValue

PropertyName = IDENTIFIER

PropertyValue = initializer

PropertyAssignment = "="
```

DEPRECATED

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

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

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

8.3.8 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.

Lifelines in DMTF sequence diagrams shall be labeled using the same format as that defined for labeling objects in DMTF object diagrams, as defined by the `InstanceLabel` rule in 8.3.7.

8.3.9 Designation of deprecated or experimental elements in diagrams

Profiles may designate profile elements as experimental (see 7.18), and revisions of profiles may deprecate profile elements defined in a previous version (see 7.19).

Profiles may refer to deprecated or experimental schema elements as part of class adaptations (see 7.13.2.1), property requirement (see 7.13.2.8), or method requirements (see 7.13.3.2).

In diagrams the depiction of respective deprecated or experimental elements, or of elements that depend on deprecated or experimental schema elements, should be designated using the following notational conventions:

- Deprecated element – suffix the letter D in curly brackets:

  {D}

- Experimental element – suffix the letter E in curly brackets:

  {E}

9 Profile implementation requirements

9.1 General

Clause 9 defines the requirements for the implementation of one or more profiles. The primary target audience for this clause is implementers of profiles.

9.2 Implementation requirements for a set of profiles

9.2.1 General

Typically, a profile is not implemented by itself but as part of the implementation of a set of profiles that is composed of one or more profiles selected by the implementer for implementation, and their referenced profiles. Such a set of profiles establishes a comprehensive management interface for a management domain that is a composition of the management domains addressed by the individual profiles.

This is also the reason why the term "implementation" (see 3.30) is defined as "a WBEM server that implements applicable portions of one or more profiles", as opposed to profile implementation (see 3.67) that is defined as "a subset of an implementation that realizes the requirements of a particular profile in a particular profile implementation context".

The term *implementation-required* is defined as follows: A profile or profile element is implementation-required if its implementation is required as part of the implementation of one or more profiles, namely

- The profile or profile element is mandatory
- The profile or profile element is conditional or conditional exclusive, and the either the condition is True, or the profile or profile element was selected to be implemented
- The profile or profile element is optional and was selected to be implemented
- The implementation type (see 7.13.2.5) is not abstract or embedded.

NOTE The implementation requirements of abstract profiles or profile elements are taken into account by concrete elements that are based on them. Likewise, the implementation requirements of embedded profile elements are taken into account by the elements embedding them.

An implementation (of a set of profiles) shall conform to the implementation requirements of these profiles and their referenced specifications.

For a functioning implementation, the following activities need to be performed:

- Determine the *implementation adaptation set* by applying the merge algorithm detailed in 9.4.

  The implementation adaptation set is composed of *implementation adaptations* (see 9.2.2).
- Implement each implementation adaptation in the implementation adaptation set, conforming to the requirements detailed in 9.3.

9.2.2 Implementation adaptation

An implementation adaptation is an adaptation that is implementation-required for a particular profile implementation. It merges the requirements of base adaptations and of other requirements sources, such as the schema definition of the adapted class, the operations specification (see 7.13.3.3.1), or of registry elements, such as alert messages or metric definitions.

An implementation adaptation does not contain requirements for optional elements that were not selected to be implemented. Such requirements are simply not merged into the implementation adaptation during processing of the merge algorithm (see 9.4).

9.2.3 Profile implementation context

It is very important to realize that a particular used profile (or, more specifically, the adaptations defined in the used profile) may need to be implemented separately for different references to that profile. The decision whether a used profile is implemented separately should be made by investigating whether the managed objects represented by adaptation instances controlled by respective profile implementations are different; if they are this is an indicator for separate profile implementations.

A profile that is not referenced by other profiles is always implemented in its own context. This is typically the case for autonomous profiles.

A profile usage may establish a separate *profile implementation context* with specific implementation requirements for the used profile; this recursively applies to profiles used by the used profile. For a particular profile implementation the profile implementation context is characterized by the chain of profile usages.

The profile implementation context can be written by stating the name of the used profile that is implemented, suffixed by the name of the using profile in parenthesis:

> If the context is a chain of profile usages, parenthesis are applied recursively. For example, a profile implementation context of "A" indicates that profile A is implemented in its own profile implementation context, a profile implementation context of "B(A)" indicates that profile B is implemented in context of an implementation of profile A, and "C(B(A))" indicates that profile C is implemented in the context of an implementation of profile B that in turn is implemented in the context of an implementation of profile A.

Figure 12 shows an example of a profile that references two other profiles, and the resulting profile implementations.

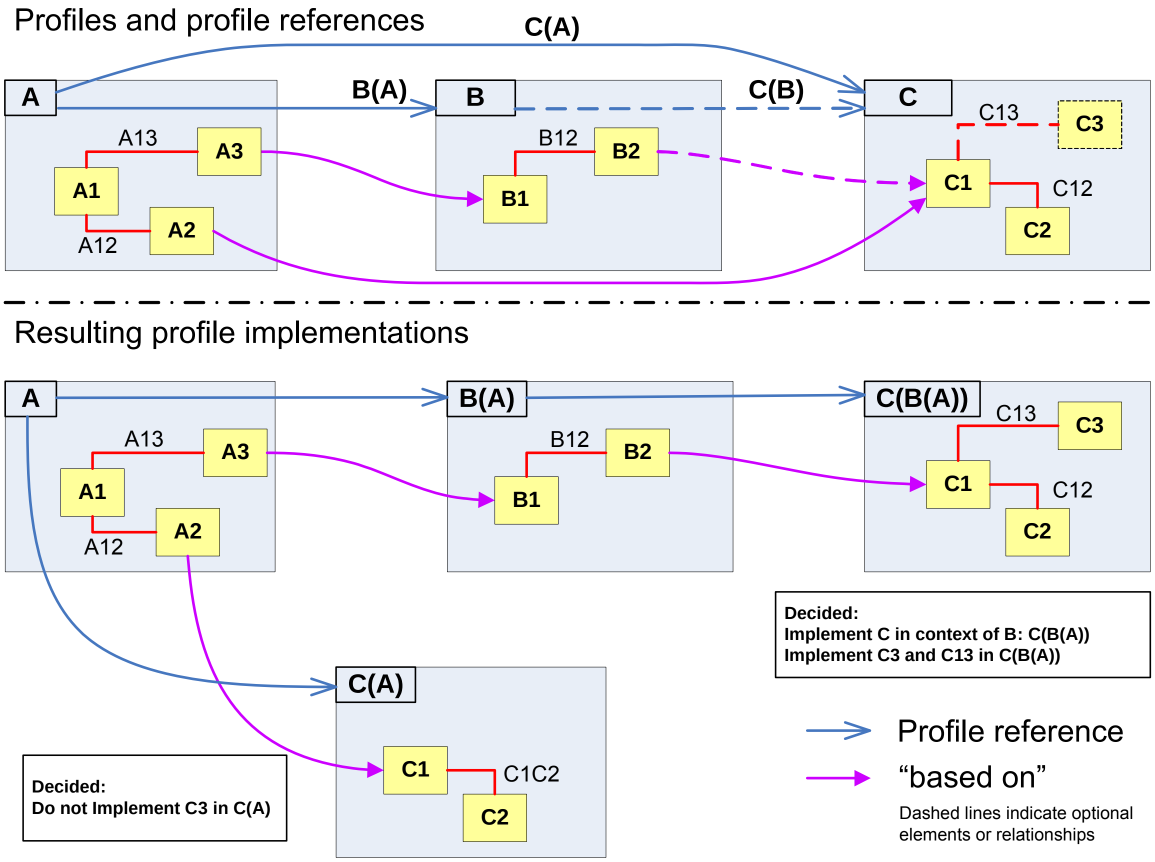

Profiles and profile references
C(A)
A
B(A)
B
C(B)
C
A13
A3
A1
A2
A12
B12
B2
B1
C13
C3
C1
C12
C2
Resulting profile implementations
A
A13
A3
A1
A2
A12
B(A)
B12
B2
B1
C(B(A))
C13
C3
C1
C12
C2
Decided:
Implement C in context of B: C(B(A))
Implement C3 and C13 in C(B(A))
C(A)
C1
C1C2
C2
Decided:
Do not Implement C3 in C(A)
Profile reference
"based on"
Dashed lines indicate optional elements or relationships

Figure 12 – Example of profiles and resulting profile implementations

The upper part of Figure 12 shows a set of profiles: Profile A references profile B and profile C as mandatory profiles, and profile B also references profile C as an optional profile.

The lower part of Figure 12 shows the resulting profile implementations in this example case: Profile A is implemented for itself because it is selected for implementation, profile B is implemented in context of profile A because it is a mandatory profile of profile A. Profile C is implemented twice — in context of profile A and in context of profile B — because it is a mandatory profile of profile A, and because it is an optional profile of profile B, and the decision was made to implement profile C in context of profile B.

In order to further substantiate the requirement for separate profile implementations, consider that adaptation C1 defined by profile C is the base adaptation for adaptation A3 defined in profile A, as well as for adaptation B2 defined in profile B. A3 as well as B2 introduce additional implementation requirements which in general are different, and can be incompatible with each other. For example, A3 might adapt a subclass of that adapted by C1, and might define property requirements for properties that are defined in that subclass, whereas B2 might define method requirements that are incompatible with those of A3.

In addition, as shown in Figure 12, for each profile implementation different decisions on optional elements are possible. For the implementation of profile C in the context of that of profile A (depicted as C(A)) it was decided not to implement adaptation C3, whereas for the implementation C(B(A)) it was decided to implement adaptation C3.

In order to distinguish implementation adaptations with different profile implementation contexts within the implementation adaptation set they need to be qualified with their profile implementation context, that is, each implementation adaptation is identified by the adaptation name and the profile implementation context.

Furthermore, for each implementation-required profile implementation, the implementation adaptations need to be constructed by merging the requirements from base adaptations.

Figure 13 shows an example of implementation adaptations that were created by merging the requirements from adaptations from the profile implementations shown in Figure 12.

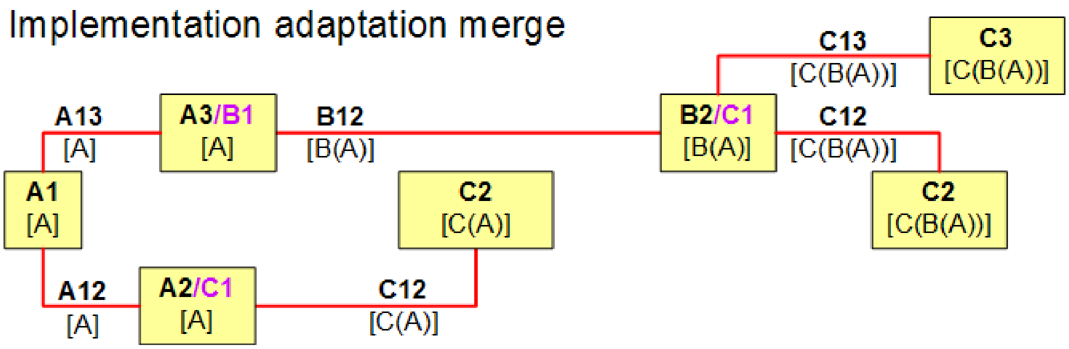


Figure 13 – Example of merging of adaptations into implementation adaptations

As shown in Figure 12, adaptation A3 defined in profile A is based on adaptation B1 defined in profile B. Figure 13 shows the result of the merge process: For example, the merge of requirements from both adaptations A3 and B1 in context of the implementation of profile A is shown as the merged implementation adaptation A3/B1[A]. Likewise, because adaptation B2 defined in profile B is based on adaptation C1 defined in profile C, the merge of requirements from adaptations B2 and C1 in context of the implementation of profile B in context of that of profile A is shown as the merged implementation adaptation B2/C1[B(A)].

Note that the profile implementation context is determined for derived adaptations that are implemented, but not for base adaptations that have an impact on those derived adaptations. For instance, in the example shown in Figure 12, profile C does not show up in the profile implementation context [B(A)] of adaptation B2/C1, even though profile C has an impact on that merged adaptation by means of base adaptation C1.

9.2.4 Implementation optimizations

During the realization of implementation adaptations optimizations are possible. Any such optimizations go beyond the scope of this guide and are mentioned for informational purposes only.

For example, if the implementation requirements do not diverge too much, it might be possible to realize two implementation adaptations with one common piece of implementing code that addresses the common requirements through a common path, and the small set of different requirements through different paths. For the example shown in Figure 13, that might be possible for C2[C(A)] and C2[C(B(A))].

An additional potential for optimization is combining instances. For example, if two or more temperature sensors have identical capabilities in all aspects (including identical temperature sensor ranges), then these capabilities could be represented by one adaptation instance. Combining instances is an optimization that is visible to clients that generally reduces the ability to represent differences and thus should be applied with great care.

9.2.5 Schema requirements

Implementations shall use the highest version of any schema from the set of schemas required by any of the profiles in the set of profiles that are implemented; beyond that, implementations should use the most recently published minor version within the same major version of any required schema.

9.3 Implementation requirements for implementation adaptations

9.3.1 General

The requirements of 9.3 apply for implementation adaptations[2] that are determined for an implementation by means of the merge algorithm detailed in 9.4.

In this subclause the implementation requirements for implementation adaptations are listed.

Keep in mind that the quantification "all" for required elements of implementation adaptations only comprises implementation-required elements (see 9.2.2). In other words, an implementation adaptation is already stripped of optional and conditional elements that were not selected or are not required to be implemented. Thus the quantification "all" each time refers to all respective elements of only the implementation adaptation, which are the implementation-required elements of the adapted class (and other implementation-required elements such as operation requirements, instance requirements and the like) that were determined by applying the merge algorithm.

For implementation adaptations with an implementation type of "instantiated", the following requirements apply:

- implement all properties[2], as detailed in 9.3.2
- implement all methods[2] and operations[2], as detailed in 9.3.3
- implement all instance requirements[2], as detailed in 9.3.4

For implementation adaptations with an implementation type of "indication", the following requirements apply:

- implement all properties[2], as detailed in 9.3.2

[2] Note that implementation adaptations are composed only of implementation-required elements; see the general remark in 9.3.1.

- implement all indication-generation requirements[2], as detailed in 9.3.5

For implementation adaptations with an implementation type of "embedded" or with an implementation type of "exception", the following requirements apply:

- implement all properties[2], as detailed in 9.3.2

9.3.2 Implementation requirements for properties

For each implementation adaptation all properties[2] shall be implemented, conforming with all value requirements and constraints established by profiles and by the schema. In particular, the profile requirements for property values to reflect the situation of the represented (aspect of the) managed object shall be implemented.

If a property is required by any of the profiles being implemented (see 9.2.1) with either the mandatory requirement level, or with the conditional or conditional exclusive requirement level and the condition being True, the property value shall not be Null when retrieved, except if specifically allowed by the profile establishing the requirement level. The non-Null value requirement does not apply for implemented optional properties.

The values of non-implemented properties shall be Null when retrieved. This is even the case if the schema definition of a property defines a non-Null default value because a schema defined default value is an initialization constraint that applies at instance creation time only.

9.3.3 Implementation requirements for methods and operations

9.3.3.1 General

For each implementation adaptation[2] with an implementation type of "instantiated" an implementation shall implement all methods[2], conforming with the method semantics defined by profiles and by the schema.

For each implementation adaptation[2] with an implementation type of "instantiated" an implementation shall implement all operations[2], conforming with the operation semantics defined by profiles and by the operations specification (see 7.13.3.3.1).

The invocation of non-implemented operations and methods shall fail, indicating that the operation or method is not implemented.

9.3.3.2 Input parameters

9.3.3.2.1 Input parameters for methods

An implementation shall implement all input parameters[2], accepting all input values as required by profiles, within the constraints and input value requirements defined by profiles and the schema. This applies likewise to property values of embedded CIM instances.

For methods the concept of optional parameters is not defined, values for all parameters are mandatory; however, Null is a valid value. Note that profiles may define specific semantics to specific values of input parameters; see 7.13.3.2.2.

If for a particular input parameter value requirements are not stated in any profile, the implementation may support all or a subset (including the case of not supporting any input value) of the admissible value set established by the schema definition of the input parameter, or in case of operations by the definition of the operation in the operations specification (see 7.13.3.3.1).

In case a value subset is supported, and if clients provide input values outside of that value subset, a respective error shall be indicated. This applies likewise to values of properties in adaptation instances provided as input.

9.3.3.2.2 Input parameters for instance creation operations

For instance creation operations the rules for implementing property values of input instances, for initializing property values that are not provided, the operation semantics and error reporting requirements are specified in the operations specification (see 7.13.3.3.1) and in profiles (see 7.13.3.3.3 and 7.13.2.11.2).

Recall that CIM instances are not created by themselves, but are the representations of (aspects of) managed objects; for details, see 6.6. Thus as part of performing an instance creation operation the implementation shall create a managed object in (or add a respective existing one to) the managed environment such that the CIM instance representing that managed object is identical to the input instance with the value determination rules applied.

If the implementation is unable to realize the instance creation in compliance with these rules, then it shall fail the instance creation operation and report a respective error.

9.3.3.2.3 Input parameters for instance modification operations

For instance modification operations the rules for implementing property values of input instances, for selecting properties for that input values are considered or disregarded, the operation semantics and error reporting requirements are specified in the operations specification (see 7.13.3.3.1) and in profiles (see 7.13.3.3.4 and 7.13.2.11.3).

Recall that modifiable CIM instances are the representations of (aspects of) managed objects; for details, see 6.6. Thus as part of performing an instance modification operation the implementation shall modify the represented managed object in the managed environment such that the CIM instance representing the modified managed object is identical to the input instance.

If the implementation is unable to realize the instance modification operation in compliance with these rules, then it shall fail the instance modification operation and report a respective error.

9.3.3.3 Output parameters

An implementation shall implement all output parameters, producing all output values within the constraints established by profiles, the schema and the operations specification (see 7.13.3.3.1), in accordance with the situation in the managed environment resulting from the method or operation execution. This applies likewise for return values.

For methods the concept of optional parameters is not defined; values for all parameters are mandatory, but Null is a legal value. For operations, optional output parameters may be defined in the operations specification, in the sense that in some situations no output values are returned.

9.3.3.4 Error reporting requirements

If error reporting requirements[2] (see 7.13.3.3.6) are defined for a method or operation, and during the method or operation execution an error occurs, the implementation shall apply the error reporting requirements that address the error situation.

An error reporting requirement is applied by sending all referenced standard error messages, and by returning the CIM status code. The CIM status code is either explicitly required as part of the error reporting requirement, or is implicitly required through the value of the CIMSTATUSCODE element of one or more of the standard error messages.

If the error situation is addressed by more than one error reporting requirement, the implementation shall apply one of those error reporting requirements, as follows:

- If a profile defines a relative order among the error reporting requirements, the implementation shall apply the error reporting requirements in that order.
- If such an order is only established by the error reporting requirements of the operations specification (see 7.13.3.3.1), the implementation shall apply the error reporting requirements in that order.
- If no order is defined, the implementation shall apply the error reporting requirements that most appropriately reports the error. The additional description provided along with the error reporting requirements may be used as a guideline for selecting for the most appropriate error reporting requirements.

9.3.4 Instance requirements

Implementations of adaptations with an implementation type of "instantiated" shall reflect the situation in the managed environment by representing (aspects of) managed objects by adaptation instances, as required by instance requirements.

9.3.5 Indication generation requirements

Implementations of adaptations with an implementation type of "indication" shall reflect the situation in the managed environment by complying with all indication-generation requirements (see 7.13.4.2), generating respective indications if the event that the indication is designed to report occurs. This applies likewise for indications reporting secondary events, such as lifecycle indications reporting changes of the CIM model as a result of prior changes in the managed environment. In addition, the requirements of the Indications profile (see DSP1054) apply.

9.4 Merge algorithm

9.4.1 General

The purpose of the merge algorithm is determining — for a set of initially selected profile implementations and their dependent profile implementations — all required implementation adaptations plus all requirements that affect that adaptation implementation, namely

- the requirements of the adapted class defined in the schema
- the requirements from the adaptation itself, namely element requirements such as property requirements, method requirements and operation requirements — both with their error reporting requirements, and the instance requirements (or — in case of indications — the indication-generation requirements)
- the respective requirements from base adaptations
- the requirements from the operations specification (see 7.13.3.3.1)
- the requirements from referenced registry elements

The merge algorithm requires the repeated processing of profile implementation checks (see 9.4.3), each requiring repeated processing of adaptation implementation checks (see 9.4.4), in order to build the implementation adaptation set.

The resulting implementation adaptation set contains — for a set of initially selected profile implementations and their dependent profile implementations — all implementation adaptations, each with all element requirements collected from the various sources listed above, and with all instance requirements or — in case of indication adaptations — indication-generation requirements.

Optimizations are possible when realizing the implementation adaptations from the implementation adaptation set; see 9.2.4.

9.4.2 Merge algorithm steps

The merge algorithm starts with step 1):

1) **Decision:** Select an initial desired set of profiles to be implemented.

2) For each profile implementation selected in step 1), perform the profile implementation check as detailed in 9.4.3, in its profile implementation context (see 9.2.3).

3) Inspect the resulting implementation adaptation set for possible implementation optimizations as described in 9.2.4.

After performing step 3), the merge algorithm is completed.

9.4.3 Profile implementation check

A profile implementation check is always to be performed in a specific profile implementation context (see 9.2.3).

1) **Decision:** Select which optional and conditional[3] features of the currently checked profile implementation are to be implemented; this will impact subsequent steps.

2) For all conditional adaptations check the condition[3], and if the condition is True, perform the adaptation implementation check (see 9.4.4), in the context of the currently checked profile implementation.

3) **Decision:** Select which optional and which conditional adaptations (with a condition of False from step 2)) of the currently checked profile implementation are to be implemented. For selected adaptations perform the adaptation implementation check (see 9.4.4), in the context of the currently checked profile implementation.

4) For base profiles of the currently checked profile implementation, perform the profile implementation check (described in this subclause), in the context of the currently checked profile implementation. This in effect causes the requirements of the base profile to be addressed as if they were requirements of the derived profile.

   NOTE Step 4) is necessary in order to pick up adaptations defined in the base profile that are not used as base adaptations, and thus require an independent implementation.

5) For all conditional profiles check the condition[3], and if the condition is True, perform the profile implementation check (described in this subclause) for the implementation of the referenced conditional profile, with the profile implementation context extended to the conditional profile.

6) **Decision:** Select which optional profiles and which conditional profiles (with a condition of False from step 5) are to be implemented. For selected profile implementations perform the profile implementation check (described in this subclause) for the implementation of the referenced optional or conditional profiles, with the profile implementation context extended to the selected optional or conditional profile.

7) **Decision:** Decide whether for the currently checked profile any scoped profiles are to be implemented. For selected profile implementations perform the profile implementation check (described in this subclause) for those profile implementations, with the profile implementation context extended to the selected scoped profile.

[3] The determination of a condition might involve optional elements. If so, at this point it needs to be decided whether these optional element(s) is (are) to be implemented, and that decision needs to be retained in later steps.

9.4.4 Adaptation implementation check

An adaptation implementation check is performed for an adaptation in a specific profile implementation context (see 9.2.3). It either creates a new implementation adaptation with that profile implementation context in the implementation adaptation set, or amends an existing one, as follows:

1) Merge the requirements as exposed by the schema definition of the adapted class. Merging means creating the implementation adaptation within the implementation adaptation set if it did not yet exist, and adding or refining the element requirements as exposed by the schema definition of the adapted class.

2) Merge the mandatory elements to the implementation adaptation (determined or created in step 1)). Merging means adding or refining the element requirements with the requirements from the adaptation defined in the profile to be implemented.

3) For any conditional elements check the condition. For those conditional elements where the condition is True, as in step 2) merge the respective element requirements to the implementation adaptation.

4) **Decision:** Select which optional and conditional elements not addressed in step 3) are to be implemented, and — as in step 2) — merge the respective element requirements to the implementation adaptation.

   NOTE The potentially complex condition check in step 3) can be avoided for those conditional elements that are selected in step 3) anyway, by performing steps 3) and 4) concertedly.

5) For any operation, merge the requirements from the operations specification (see 7.13.3.3.1).

6) If the subject adaptation is based on other adaptations, perform the adaptation implementation check (described in this subclause) for the direct base adaptations, using the profile implementation context of the profile defining the subject adaptation, and then — in the context of the profile defining the base adaptation — mark the implementation of the direct base adaptations as addressed by a derived adaptation. The last part is necessary in order to avoid picking up those requirements in a later execution of step 4) of the profile implementation check.

9.5 Implementation of deprecated definitions

Implementations shall conform to definitions of the schema, profiles and the operations specification (see 7.13.3.3.1) regardless of whether or not they are deprecated. Clients should not rely on or exploit deprecated definitions, and they are encouraged to stop exploiting deprecated functionality as soon as possible.

10 Profile specification requirements

10.1 General

Clause 10 defines the requirements for profile specifications. Profile specifications are documents containing the definition of one or more profiles in textual form.

Clause 10 focuses on formal text document aspects. In addition, all requirements stated in clause 7 for profile definitions and the general conventions and guidelines for profile defined in clause 8 apply to profile specification documents.

A profile specification published by DMTF shall conform to all requirements of this guide; in addition the requirements of ISO/IEC Directives, Part 2 apply. The conformance requirements for profiles and profile specifications are detailed in clause 5.

10.2 Profile specification conventions

10.2.1 Conventions for the specification of requirement levels

In profile specifications, requirement levels (see 7.3) are stated using keywords as defined in this subclause.

- The mandatory requirement level (see 7.3.2) shall be stated using the keyword "mandatory".
- The conditional requirement level (see 7.3.4) shall be stated using the keyword "conditional"; in addition, the requirements described in 10.2.3 for the specification of the condition apply.
- The conditional exclusive requirement level (see 7.3.5) shall be stated using the keyword "conditional exclusive"; in addition, the requirements described in 10.2.3 for the specification of the condition apply.
- The optional requirement level (see 7.3.3) shall be stated using the keyword "optional".
- The prohibited requirement level (see 7.3.6) shall be stated using the keyword "prohibited".

10.2.2 Conventions for the specification of implementation types

In profile specifications, the implementation types (defined for adaptations, see 7.13.2.5) are stated using keywords as defined in this subclause.

- The "instantiated" implementation type shall be stated using the keyword "instantiated".
- The "embedded" implementation type shall be stated using the keyword "embedded".
- The "abstract" implementation type shall be stated using the keyword "abstract".
- The "indication" implementation type shall be stated using the keyword "indication".
- The "exception" implementation type shall be stated using the keyword "exception".

10.2.3 Conventions for the specification of conditional elements

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

10.2.3.1 General

Conditions shall be defined using one of the mechanisms defined in 7.4.

10.2.3.2 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 " ConditionalFlavor "."

ElementName = PROFILE_IDENTIFIER / IDENTIFER ; shall identify the conditional element

ElementType = "profile" / "feature" / "adaptation" / "property" / "method"
/ "parameter"

ConditionalFlavor = "conditional" / "conditional exclusive"
```

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 the English language.

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

```
ConditionStatement = "Condition:" *WSP ConditionSpecification
```

`ConditionSpecification` shall be an appropriate textual representation of the basic types of conditions and their combination using Boolean operators, as specified in 7.4.

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."

10.2.3.3 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" / "Conditional exclusive"
```

The condition shall be specified in a corresponding cell within the Description column of the same table. If the text in the Description cell would exceed a reasonable amount of words (about 20 words), it shall be replaced by a reference to a separate subclause that defines the condition, following the conventions defined in 10.2.3.2.

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

10.2.4 Conventions for the specification of value constraints

As defined in 7.13.2.10, a profile may constrain property values or method parameter values to a single value or a set of values. Also, for string-typed properties, methods and parameters, profiles may specify a mechanism that conveys the format used for their values.

In profile specifications, value constraints may be expressed in the form of ABNF, or in the form of a regular expression. This subclause details conventions to be applied if regular expressions are used.

Table 3 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, it 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 3.

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 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 3.
- For the specification of the 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 3.
- 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 3.
- 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 the `ValueMap` value set and followed by the parenthesized corresponding (textual) element of the `Values` value set. For an example, see `PortType` in Table 3.
- 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 3.

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 (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 the form of normative sentences, 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 3 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 3 – 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 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 | 8 | 9 (SNVM or NodeWWN or 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 system's 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 system's Fibre Channel WWN and shall be formatted as 8 consecutive uppercase hex digits (pattern "^[0123456789ABCDEF]{8}$").

```
…
```

10.2.4.1 Conventions for the specifications of default property values

If a profile defines a default value for a property (see 7.13.2.9), that shall be specified using the following format:

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

10.2.4.2 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 7.13.2.8.

The format is

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

DEPRECATED

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

`MultiplicityValue` shall specify the multiplicity, as follows:

`"1"` indicates that exactly one instance is referenced

`"*"` indicates that 0 or more instances are referenced

`"m..n"` indicates that `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.

10.3 Profile specification structures

10.3.1 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 they 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.

10.3.2 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 for each adaptation.

DEPRECATED

10.3.3 Traditional profile specification structure

10.3.3.1 General

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 "Methods" clauses that provide detail information.

With version 1.1 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.

10.3.3.2 Specific requirements for DMTF class diagrams in traditional profile specifications

The requirements in this subclause apply in addition to those specified in 8.3.6.

Each profile specification in profile specifications applying the traditional profile structure shall contain one DMTF profile class diagram that depicts the central elements of the management interface defined by the subject profile by showing profiled classes and associations defined by the subject profile or by a referenced profile (see 7.9). That DMTF profile class diagram shall have a label formatted as follows:

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

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

Profile classes defined by the subject profile shall be represented with a box that exhibits two horizontal compartments.

The top compartment shall contain the "profile class" name as defined in 7.13, 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 " ProfileDesignator ")"
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 "profile classes" 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 adaptations 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, which 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

10.3.4 Usage of profile specification structures

The two profile specification structures are depicted in Figure 14.

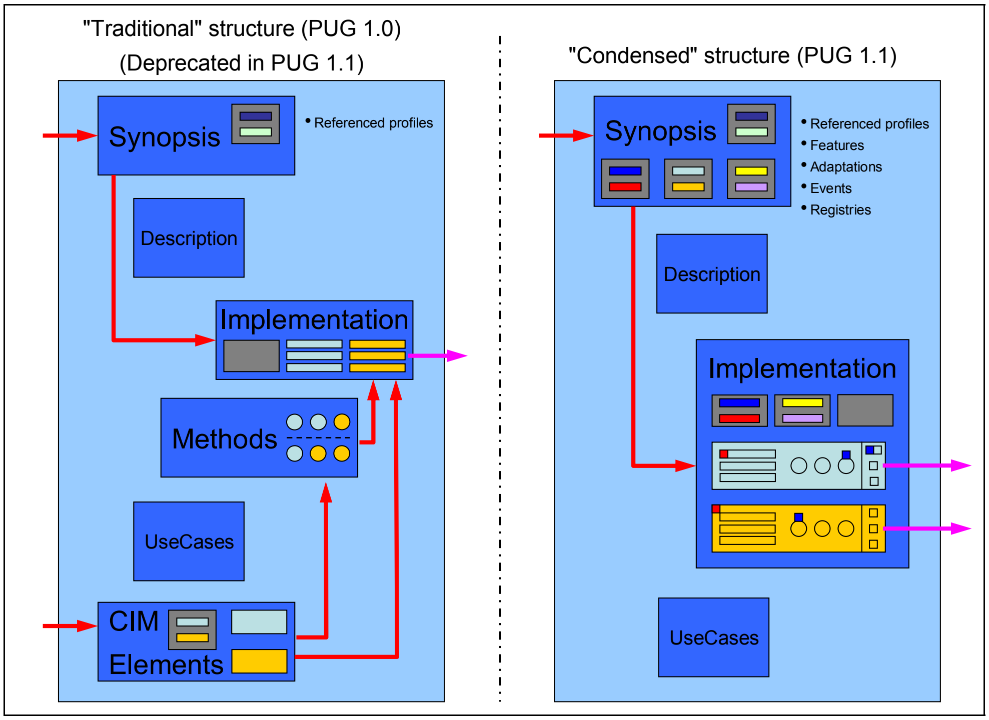


Figure 14 – Traditional and condensed profile structures

On the left side of Figure 14, the major clauses are shown with the traditional profile specification structure applied. Note the two entry paths into the profile, one following through the "Synopsis" clause, and the other one following through the "CIM elements" clause.

On the right side of Figure 14, 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 required only as the "blue" or the "red" features are implemented.

10.4 Requirements for profile specification clauses

10.4.1 General

The requirements for profile specification clauses differ with the structure chosen for the subject profile; see 10.3. Table 4 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 4 – Requirements for profile specification clauses

Clause name	Condensed structure	Traditional structure
Scope	Required, see ISO/IEC Directives, Part 2, 6.2.1.	

Normative references	Required, see ISO/IEC Directives, Part 2, 6.2.2.	
Terms and definitions	Required, see 10.4.3 and ISO/IEC Directives, Part 2, 6.3.1.	
Symbols and abbreviated terms	Required, see ISO/IEC Directives, Part 2, 6.3.2.	
Conformance	Optional, see 10.4.4.	
Synopsis	Required, see 10.4.3. Requirements differ based on the chosen structure.	
Description	Required, see 10.4.6.	
Implementation	Required, see 10.4.7. Requirements differ based on the chosen structure.	
Methods	Prohibited, content covered in "Implementation" clause; see 10.4.7.	Required, see 10.4.8.
Use cases	Required, see 10.4.9.	
CIM elements	Prohibited, content covered in "Implementation" clause; see 10.4.7.	Required, see 10.4.10.

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

10.4.2 Requirements for the numbering of profile specification clauses and subclauses

ISO/IEC Directives, Part 2 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 specification format for readers.

10.4.3 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, Part 2, 6.3.1 and Appendix D.

NOTE ISO/IEC Directives, Part 2 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 5 immediately after the heading.

Table 5 – Common text for the "Terms and definitions" clause of profile specifications

The verbal phrases "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, Part 2, Annex H. The verbal phrases in parenthesis are alternatives for the preceding verbal phrase, for use in exceptional cases when the preceding verbal phrase cannot be used for linguistic reasons. Note that ISO/IEC Directives, Part 2, Annex H specifies additional alternatives. Occurrences of such additional alternatives shall be interpreted in their normal English meaning.

The terms "clause", "subclause", "paragraph", and "annex" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 5.

The terms "normative" and "informative" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, 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, DSP0223 and DSP1001 apply to this profile.

10.4.4 Requirements for the specification of the "Conformance" clause

The specification of a conformance clause is optional.

Generally, the conformance definitions defined by this guide (see clause 5) apply.

Profiles may specify additional conformance rules for implementations beyond those required in 5.2; this guide does not define rules on how to define such conformance rules in profiles.

10.4.5 Requirements for the specification of the "Synopsis" clause

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

10.4.5.1 General

Each profile specification shall have a "Synopsis" clause.

The "Synopsis" clause of a profile specification shall conform to the rules defined in subclauses 10.4.5.4 to 10.4.5.8.

10.4.5.2 Requirements for the sequence of definitions in the "Synopsis" clause

The definitions in the "Synopsis" clause shall be in the following sequence:

- the profile attributes, as defined in 10.4.5.4
- the summary, as defined in 10.4.5.5
- the table of profile references, as defined in 10.4.5.6
- the tables of registry references, as defined in 10.4.5.7
- the table of features, as defined in 10.4.5.8
- the table of adaptations, as defined in 10.4.5.9
- the table of use cases, as defined in 10.4.5.10

Some of these definitions are only required if the corresponding elements are defined in the profile, and some are placed elsewhere when the traditional structure is used by the profile specification; this is detailed in the referenced subclauses.

10.4.5.3 Requirement for separate subclauses within the "Synopsis" clause

NOTE ISO/IEC Directives, Part 2 requires that no normative text be put at the beginning of a clause if that clause contains subclauses (to avoid "hanging" paragraphs); this is the reason for requiring separate subclauses in the case that 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 10.4.5, or because the definition of the scoping algorithm requires a separate subclause.

Consequently, if any of the definitions within the "Synopsis" clause of a profile specification requires a separate subclause, then each of the definitions listed above needs to be put in a separate subclause within the Synopsis clause.

10.4.5.4 Requirements for the specification of profile attributes

10.4.5.4.1 General

If the profile attributes are specified in a separate subclause within the "Synopsis" clause (see 10.4.5.3), that subclause shall be named "Profile attributes".

Profile attributes shall be listed as a sequence of attribute statements. This sequence of statements should be placed first in the "Synopsis" clause.

The sequence of attribute statements and their format in ABNF is defined by the "Attribute statement" column of Table 6; corresponding values in the "Requirements" column refer to subclauses of clause 7 that provide details about the respective profile attributes. In a profile specification the sequence of attribute statements should not be formatted as a table, but as a contiguous sequence of attribute value statements that are in the sequence and format detailed in Table 6.

Table 6 – Requirements for the specification of profile attributes

Attribute statement (ABNF)	Requirement
`"Profile name:" WS RegisteredProfileName` `RegisteredProfileName` shall be the registered profile name; see 7.6.2.	Required.
`"Version:" WS RegisteredProfileVersion` `RegisteredProfileVersion` shall be the registered profile version; see 7.6.3.	Required.
`"Organization:" WS RegisteredOrganizationName` `RegisteredOrganizationName` shall be the registered organization name; see 7.6.4.	Required.
`"Abstract indicator:" WS AbstractProfileIndicator` `AbstractProfileIndicator` shall be `"True"` for abstract profiles (see 7.10.1), and `"False"` otherwise. Default: `"False"`.	Required for abstract profiles.
`"Profile type:" WS ProfileType` `ProfileType` shall be `"autonomous"` for autonomous profiles (see 7.8.2), and `"component"` for component profiles (see 7.8.3).	Required.
`"Schema name:" WS SchemaName` `SchemaName` shall be the schema name; see 7.7.3. Default: `"CIM"`.	Optional.
`"Schema version:" WS SchemaVersion` `SchemaVersion` shall be the schema version; see 7.7.2.	Required unless "Schema:" is used.

For experimental schemas, the value should be suffixed with "(Experimental)"	
"Schema organization:" WS SchemaOrganization SchemaOrganization shall be the schema organization; see 7.7.4. Default: "DMTF".	Optional .
"Schema:" WS [SchemaOrganization WS] SchemaName *WS SchemaVersion SchemaOrganization, SchemaName and SchemaVersion shall be set as defined above in this table. Alternative to the specification of the triplet "Schema name", "Schema version" and "Schema organization" that should be preferred if multiple schemas are referenced.	Optional.
"Central class adaptation:" WS CentralClassAdaptationName CentralClassAdaptationName shall be the name of the central class adaptation; see 7.9.3.2.	Required.
"Scoping class adaptation:" WS ScopingClassAdaptationName ScopingClassAdaptationName shall be the name of the scoping class adaptation; see 7.9.3.3.	Required for component profiles.
"Scoping algorithm:" WS ScopingPath For ScopingPath, see 10.4.5.4.2.	Required for component profiles.
NOTE Profile attributes shall be listed in normal text font, with the profile attribute names (the initial literal up to and including the colon) highlighted in bold font; see also the example in A.2.	

10.4.5.4.2 Scoping path

ScopingPath shall be the scoping path; see 7.9.3.4. It shall be specified as follows:

- If the scoping path between central class adaptation and scoping class adaptation is composed of only one association adaptation, ScopingPath shall be the name of the association adaptation.
- Otherwise, the definition of the scoping path 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, ScopingPath shall have the form "See " SubclauseNumber, where SubclauseNumber 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.

10.4.5.5 Requirements for the specification of the summary

If the summary is specified in a separate subclause within the "Synopsis" clause (see 10.4.5.3), that subclause shall be named "Synopsis".

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 adaptations.

If the subject profile is an abstract profile, the following statement shall be included as the 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 summary is provided in A.2.

10.4.5.6 Requirements for the specification of the table of profile references

If the table of profile references is specified in a separate subclause within the "Synopsis" clause (see 10.4.5.3), that subclause shall be named "Profile references".

If the subject profile references other profiles, the requirements for profile references shall be listed in a table of profile references, as defined in this subclause. In that table each profile reference shall conform to the requirements in 7.9.

The table of profile references shall be labeled: "Profile references". In Table 7, requirements for columns in the table of profile references are defined. Each required column is described by an entry in the list provided in Table 7. 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 7 – Requirements for columns of the table of profile references

Profile reference name – Cell values shall state the name of the profile reference within the subject profile; see 7.9.1.

Profile name – Cell values shall state the registered name of the referenced profile; see 7.9.1.3.

Organization – Cell values shall state the registered organization of the referenced profile; see 7.9.1.3.

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; see 7.9.1.3.

Relationship – Cell values shall state the type of the profile reference; see 7.9.1.2.

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 referenced profile in the context of the subject profile.

– For conditional profiles the condition shall be specified using one of the mechanisms specified in 7.4.

– If the text in the "Description" cell would exceed a reasonable amount of words (about 20 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." In this case, the table shall not be included.

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

10.4.5.7 Requirements for the specification of the tables of registry references

If the tables of registry references are specified in a separate subclause within the "Synopsis" clause (see 10.4.5.3), that subclause shall be named "Registry references".

If the subject profile references message registries, the message registry references shall be listed in a table of message registry references, as defined in this subclause. The table of message registry references shall be labeled: "Message registry references".

If the subject profile references metric registries, the metric registry references shall be listed in a table of metric registry references, as defined in this subclause. The table of metric registry references shall be labeled: "Metric registry references".

In Table 8 requirements for columns in tables of registry references 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 tables of registry references

Registry reference name – Cell values shall state the name of the registry reference within the subject profile; see 7.9.1.

Registry identifier – Cell values shall state the identification of the referenced registry; see 7.12.

Organization – Cell values shall state the name of the organization that owns the referenced registry; see 7.12.

Version – Cell values shall state the version of the referenced registry; see 7.12.

Description – Cell values should provide a description of the use of referenced registry within the subject profile; see 7.12.

The following rules apply:

– If the value in any Description cell would exceed a reasonable amount of words (about 20 words), a separate subclause shall be provided within the "Implementation" clause, and the description shall be provided as part of that separate subclause. The separate subclause shall be referenced from the table entry, as follows:

```
"See" WS SubclauseNumber "."
```

`SubclauseNumber` is the number of the separate subclause.

10.4.5.8 Requirements for the specification of the table of features

If the table of features is specified in a separate subclause within the "Synopsis" clause (see 10.4.5.3), that subclause shall be named "Features".

If the subject profile defines features (see 7.15), 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 in compliance to version 1.0 of this guide) by upgrading to version 1.1 of this guide. However, note that the upgrade may require minor formal adjustments of the original version to comply with version 1.1 of this guide.

The table of features shall be labeled: "Features". In Table 9 requirements for columns in tables of features are defined. Each required column is described by an entry in the list provided in Table 9. 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 table of features

Feature name – Cell values shall state the name of the feature; see 7.15.3.

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 conditional exclusive, the cell value shall be "Conditional exclusive".

− If the feature is optional, the cell value shall be "Optional".

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

The following rules apply:

– The feature definition subclause in the "Implementation" clause (see 10.4.7.3) shall be referenced. No other text should be added.

If the specified profile does not define features, the following text shall be stated: "No features are defined in this profile." In this case, the table shall not be included.

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

10.4.5.9 Requirements for the specification of the table of adaptations

The adaptations (see 7.13) defined in the subject profile shall be listed in a table of adaptations.

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 adaptations shall be specified in the "Overview" subclause of the "CIM elements" clause (see 10.4.10.2), and the requirements for a table of adaptations as part of the "Synopsis" clause as specified in the remaining part of this subclause do not apply.

If the condensed profile specification structure is applied by the subject profile, a table of adaptations shall be specified as part of the "Synopsis" clause. All class adaptations (including the adaptations of ordinary classes, of association classes, and of indication classes) defined by the subject profile shall be listed in the table of adaptations.

If the table of adaptations is specified in a separate subclause within the "Synopsis" clause (see 10.4.5.3), that subclause shall be named "Adaptations".

The table of adaptations shall be labeled: "Adaptations". In Table 10, requirements for columns in the table of adaptations are defined. Each required column is described by an entry in the list provided in Table 10. 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 table of adaptations

Adaptation – Cell values shall state the name of the adaptation; see 7.13.

The following rules apply:

– If an adaptation is based on other 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 pertaining to elements of one adaptation are specified in separate subcells that are spanned by the cell in the "Adaptation" column.

The following rules apply:

- The first subcell shall contain the name of the adapted class.
- If base adaptations are defined, these may be stated in subsequent subcells. This should only be done for adaptations that are not described in a separate adaptation-specific subclause, as detailed with the rules for the Description column.

  The following ABNF defined format applies:

  ```
  AdaptationReference = [ ProfileName "::" ] AdaptationName
  ```

  If a base adaptation is defined in a referenced profile, then `ProfileRefName` shall be the profile reference name (see 7.9.1). `AdaptationName` shall be the name of the base adaptation

Requirement – Cell values shall state the requirement level for the adaptation; see 10.2.1.

The following rules apply:

- If an adaptation is based on other 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 base adaptations listed in 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.
- If the implementation type (see 7.13.2.5) of an adaptation is "abstract", the cell shall contain a statement indicating that the requirement level is defined in derived adaptations.

Description – Cell values shall provide a description of the adaptation.

The following rules apply:

- Unless fitting into a reasonable space within the table cell (about 20 words), the adaptation description should be provided in a separate subclause of the "Adaptations" subclause within the "Implementation" clause; see 10.4.7.4.3. The adaptation specific subclause shall be referenced from the table entry, as follows:

  ```
  "See" AdaptationSubclauseNumber "."
  ```

  `AdaptationSubclauseNumber` shall be the number of the adaptation-specific subclause.
- If the description is provided within the table cell, it shall state the implementation type.
- If no requirements are defined beyond those defined in the schema definition of the adapted class, this may be indicated by the phrase:

  ```
  "See CIM schema definition."
  ```
- If present, the subcells for the descriptions of base adaptations shall contain a reference to the subclause or profile defining the base adaptation, as follows:

  ```
  "See " BaseReference "."
  ```

  where `BaseReference` either refers to the subclause that describes the base adaptation, or is the internal document reference to the profile that defines the base adaptation.

The adaptation table shall be subdivided into two table sections that are named as follows:

- "Instantiated and embedded class adaptations"
- "Indications and exceptions"

Each table section shall be preceded by a row that spans all columns and contains the section name. The table sections shall contain the entries for adaptations defined by the profile with respective implementation types (see 7.13.2.5).

The sequence in which adaptations are listed within each of these table sections is not defined in this guide. Profiles may use any reasonable approach for that, for example an alphabetical sequence or an order implied by dependencies of the adaptations. Also, the sequence as listed in the table of adaptations may differ from the sequence of referenced adaptation-specific subclauses (see 10.4.7.4).

If a profile does not define adaptations for indications and/or exceptions, the table still shall contain the "Indications and exceptions" table section, with one entry stating that no adaptations for indications or exceptions are defined.

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

10.4.5.10 Requirements for the specification of the table of use cases

A table of use cases is only required if the condensed profile specification structure is applied by the subject profile.

In this case, the table of use cases shall be specified as part of the "Synopsis" clause. All use cases defined by the subject profile within the "Use cases" clause (see 10.4.9) shall be listed in the table of use cases.

If the table of use cases is specified in a separate subclause within the "Synopsis" clause (see 10.4.5.3), that subclause shall be named "Use cases".

The table of use cases shall be labeled: "Use cases". In Table 11 requirements for columns in the table of use cases 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 the table of use cases

Use case – Cell values shall state the name of the use case; see 10.4.9.3.1.

Description – Cell values shall refer to the subclause within the "Use cases" clause that describes the use case; see 10.4.9.3.

An example of a table of use cases is provided in A.2.

10.4.6 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 which the subject profile defines adaptations.
- should contain some or all of the following diagrams that detail the purpose of the subject profile:
  - The "Description" clause of profile specifications written in conformance with the condensed structure (see 10.3.2) should contain one or more DMTF collaboration structure

diagrams (see 8.3.4) that detail the collaboration defined by the subject profile, or should contain one or more DMTF adaptation diagrams (see 8.3.5).

Each adaptation defined by the subject profile should appear at least once in these diagrams.

– The "Description" clause of profile specifications written in conformance with the traditional structure (see 10.3.3) should contain one or more DMTF profile class diagrams (see 10.3.3.2) that detail the model defined by the subject profile.

– The "Description" clause may contain DMTF object diagrams (see 8.3.7) providing details on CIM instances, their interactions, and their relationship to managed objects in managed environments, as required by the subject profile.

Table 12 lists the requirements for diagrams as part of the Description clause within profile specifications. Note that the requirements depend on the structure chosen for the profile specification; see 10.3.

Table 12 – Profile diagram types

Diagram type	**Usage requirements**		**Description**
	Traditional structure	**Condensed structure**	
DMTF collaboration structure (EXPERIMENTAL)	Optional	Optional.	See 8.3.4.
DMTF class adaptation (EXPERIMENTAL)	Optional	Required, unless a DMTF collaboration structure diagram is shown.	See 8.3.5.
DMTF class	Not defined	Optional	See 8.3.6.
DMTF profile class (DEPRECATED)	Required, unless the profile revision was changed to specifying adaptations in place of "profile classes". In this case a DMTF collaboration structure or a DMTF class adaptation diagram is required.	Not applicable	See 10.3.3.2.
DMTF object	Optional	Optional	See 8.3.7.
DMTF sequence	Optional	Optional	See 8.3.8.

An example of a "Description" clause is provided in A.3.

10.4.7 Requirements for the specification of the "Implementation" clause

This subclause defines requirements for the "Implementation" clause in profile specifications.

10.4.7.1 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 (direct and indirect) base profile(s), the "Implementation" clause shall only contain the statement "All implementation requirements are defined in base profile(s)."

10.4.7.2 Usage of subclauses

The "Implementation" clause should be structured into subclauses.

Subclauses may introduce subtopics that apply to one or more profile elements (for example a subclause titled "Element discovery"), or they may introduce subtopics that address specific profile elements (for example, a specific 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 10.4.7.3
- A required "Adaptations" subclause, as detailed in 10.4.7.4

NOTE ISO/IEC Directives, Part 2 requires 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, Part 2 would require another subclause of the "Implementation" clause. In this case, an initial subclause named "General" containing general definitions is recommended.

10.4.7.3 Requirements for the specification of features

If the subject profile defines features (see 7.15), 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 subclause shall be formatted as follows:

```
FeatureSubclauseTitle = "Feature: " FeatureName
```

The value of `FeatureName` shall be the name of the feature; see 7.15.3.

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 10.2.3.

Each feature definition subclause shall provide all of the following (in the order stated):

- A description of the feature
- The granularity of the feature; see 7.15.5
- The requirement level of the feature; see 7.15.4
- A description of one or more discovery mechanisms for the feature; see 7.15.6.

The implementation requirements that result from a decision to implement a feature are not defined as part of the feature definition subclause; see 7.15.7.

10.4.7.4 Requirements for the specification of adaptations

This subclause defines requirements for the specification of adaptations, addressing the requirements of 7.13.

10.4.7.4.1 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 adaptation (including adaptations of association classes or indication classes) defined by the profile as specified in 10.4.7.4.3, unless the 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 adaptations based on them. The description of a trivial class adaptation may be solely provided in the entry in the table of adaptations within the "Synopsis" clause if the space requirements for table cells are met; see 10.4.5.9.

The sequence in which adaptation-specific subclauses appear in the "Adaptations" subclause is not defined in this guide. Profiles may use any reasonable approach for that, for example an alphabetical sequence or an order implied by dependencies of the adaptations. Also, the sequence as listed in the table of adaptations (see 10.4.5.9) may differ from the sequence of referenced adaptation-specific subclauses.

10.4.7.4.2 Requirements for the specification of conventions

The "Adaptations" subclause of the "Implementation" clause shall contain a subclause named "Conventions" that specifies the conventions applied within the profile specification for the definition of adaptations. The "Conventions" subclause shall precede any subclause defining adaptations.

This guide requires profiles to repeat certain schema requirements (see 7.13.2.8.3). Within a profile specification, in these cases the convention shall be to state the name of the qualifier if its effective value is True, and to not state the name of the qualifier if its effective value is False. This convention shall be applied for the Key and the Required qualifiers as part of property requirements as required by 7.13.2.8.3 and as detailed in 10.4.7.4.3, and for the In, Out, and Required qualifiers as part of method parameter requirements as detailed in 10.4.7.4.6. If applied anywhere in a profile specification, this convention shall explicitly be stated as part of the "Conventions" subclause, along with a brief description of what the respective qualifier value means.

This guide requires profiles to select DSP0223 as the operations specification that defines the operations for that the profile defines operation requirements; see 7.13.3.3.1. Profiles are required to specify operation requirements individually per adaptation (see 10.4.7.4.7). This requirement shall be stated in the form of a respective convention within the "Conventions" subclause.

An example of an adaptation related "Conventions" subclause is provided in A.4.3.

10.4.7.4.3 Requirements for the specification of individual adaptations

Each adaptation definition subclause within the "Adaptation" subclause of the "Implementation" clause shall be titled

```
    AdaptationClauseTitle = [ "Adaptation" [ *WSP ] ":" *WSP ] AdaptationName
    [ *WSP ] ":" *WSP AdaptedClassName
```

`AdaptationName` is the name of the adaptation (see 7.13.2), and `AdaptedClassName` is the name of the adapted class.

Each adaptation-specific subclause shall define implementation requirements. Implementation requirements may be defined directly within the adaptation-specific subclause, or within separate subclauses.

Each adaptation-specific subclause shall state the implementation type of the adaptation (see 7.13.2.5).

Requirements for elements of adaptations, such as base adaptations, alert messages, metrics, properties, methods, and operations, shall be stated in the form of an "Element requirements" table. In that table each entry shall be assigned a requirement level. If needed, the table entries may refer to other subclauses that provide detail information.

NOTE Implementation requirements may also be imposed from other sources, such as the schema or the operations specification. Clause 9 details a merge algorithm that produces a set of implementation adaptations, merging the implementation requirements from those various sources.

The "Element requirements" table listing required elements of the adaptation shall be labeled:

```
ElementRequirementsTableTitle = AdaptationName [ *WSP ] ":" *WSP "Element
requirements"
```

`AdaptationName` is the name of the adaptation (see 7.13.2).

Table 13 defines requirements for columns in adaptation element tables. Each required column is described by an entry in the list provided in Table 13. 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 13 – Requirements for columns of "Element requirements" tables

Element – Cell values shall state the name of the base element, property, method, or operation, or the identification of a metric for which the subject profile defines requirements as part of the defined adaptation.

The following rules apply:

– If base adaptations are defined, these shall be stated, using the following format:

```
AdaptationReference = [ ProfileRefName "::" ] AdaptationName
```

If a base adaptation is defined in a referenced profile, then `ProfileRefName` shall be the profile reference name (see 7.9.1). `AdaptationName` shall be the name of the base adaptation.

– If an alert indication adaptation refers to one or more alert messages defined in a message registry (see 7.13.4), the identifier of the alert message shall be stated, using the following format:

```
MessageIdentification = MessageRegistryRefName "::" MessageID
```

`MessageRegistryRefName` shall be the message registry reference name (see 7.12) of the registry in which the message on which the indication is based is defined, and `MessageID` shall be the message id of that message. The message id is the concatenation of the value of the PREFIX attribute and the SEQUENCE_NUMBER attribute from the MESSAGE_ID element that describes the message in the message registry.

– Array property names shall be suffixed with "`[ ]`".

– Method names and operation names shall be suffixed with "`( )`".

– Names of association traversal operations (see 10.4.7.4.8) shall be specified as follows:

```
OpName "( )" [ " WS "for" WS AssocAdaptationSet ]
```

where `OpName` is the operation name, as defined by the operations specification (see 7.13.3.3.1).

If the "for" suffix is not specified, the operation requirement affects all association adaptations specified by the subject profile that reference the adaptation defined in the subclause containing the table.

If the "for" suffix is specified, the operation requirement affects a subset of the association adaptations specified by the subject profile that reference the adaptation defined in the subclause containing the table. In this case, `AssocAdaptationSet` shall list that subset, as follows:

```
AssocAdaptationSet = AssocAdaptation [ *WSP "," *WSP AssocAdaptationSet ]
```

`AssocAdaptation` shall identify an association adaptation specified by the subject profile that references the adaptation defined in the subclause containing the table.

– Identifications of metric-defining metric requirements shall be stated using the following format:

```
MetricReference = MetricRegistryRefName [ *WSP ] "::" *WSP METRICID
```

`MetricRegistryRefName` is the name of the metric registry reference that references the metric registry within that the metric for the metric requirement is defined, and `METRICID` identifies the metric within the metric registry, as defined in DSP8020.

Requirement – Cell values shall state the requirement level of the element requirement.

– The requirement level shall be stated in conformance to the conventions defined in 10.2.1.

– For property requirements, the presentation requirement level (see 7.3.1) shall be stated.

– If the profile allows the value Null for the property (see 7.13.2.10.4), the requirement level may be amended, as follows:

```
Requirement = RequirementLevel *WSP "," *WSP "NullOK"
```

`RequirementLevel` is the requirement level stated in conformance to the conventions defined in 10.2.1.

– If a property requirement also contains property initialization value requirements (see 7.13.2.11.2) and/or property modification value requirements (see 7.13.2.11.3), these shall be placed into a separate subclause that is referenced in by the value in the "Description" cell (as detailed under "Description").

Description – Cell values shall conform to the following specifications:

The following rules apply:

– Repetition of the effective qualifier values from the schema definition of the adapted class:

  – The convention requirements defined in 10.4.7.4.2 apply.

  – If the effective value of the Key qualifier is True for a property, the word "Key" shall be listed first in the description of the property requirements; if the effective value is False, the name of the qualifier shall not be listed.

  – If the effective value of the Required qualifier is True for a property, the word "Required" shall be listed first in the description of the property requirements; if the effective value is False, the name of the qualifier shall not be listed. Note that the meaning of the Required qualifier is that the value of the qualified element shall not be Null.

  – If both qualifiers have the effective value True, their names shall be presented in the form of a comma separated list.

– If the requirement level is "conditional" or "conditional exclusive", and unless the condition is already stated in the "Requirement" column, the condition shall be stated here, as detailed in 10.2.3.

– The managed object type that is modeled by the adaptation.

– The definition of additional requirements shall be stated, as follows:

  – Property requirements shall be specified as detailed in 10.4.7.4.4.

  – Method requirements shall be specified as detailed in 10.4.7.4.6.

  – Operation requirements shall be specified as detailed in 10.4.7.4.7 and 10.4.7.4.8.

– The keyword "Deprecated" shall be stated if the required element is marked deprecated by the profile, in the schema definition or in the operations specification (see 7.13.3.3.1); for details, see 7.19.

If present, and if defined in the subject profile, the cell for the description of a base adaptation shall contain a reference to the subclause defining the base adaptation, as follows:

```
"See " SubclauseNumber "."
```

where `SubclauseNumber` is the number of the subclause containing the definition of the base adaptation.

If defined in a referenced profile, the cell for the description of a base adaptation shall contain a reference to the referenced profile defining the base adaptation, as follows:

```
"See " ProfileReference "."
```

where `ProfileReference` is the internal document reference to the profile that defines the base adaptation.

– If present, the cell for descriptions of an alert message should contain a reference to the message registry defining the alert message, as follows:

```
"See " MessageRegistryReference "."
```

where `MessageRegistryReference` is the internal document reference to the message registry that defines the alert message.

– Unless fitting into a reasonable space within the table cell (about 20 words), the element description should be placed in a separate subclause of the adaptation-specific subclause, and referenced from the table cell.

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.

Depending on the presence of respective requirements, adaptation element tables shall be subdivided into table sections. Each table section shall be preceded by a row that spans all columns and contains the section name. The following conventions should be applied:

- If base adaptations are defined, these should be listed in a table section named `Base adaptations`
- If alert messages are referenced as part of an alert indication adaptation, the alert message references should be listed in a table section named `Alert messages`
- If metric definitions are referenced as part of a adaptation defining metric requirements, the metric definition references should be listed in a table section named `Metrics`
- If property requirements are defined, these should be listed in a table section named `Properties`
- If method requirements are defined, these should be listed in a table section named `Methods`
- If operation requirements are defined, these should be listed in a table section named `Operations`

Requirements for optional properties, methods, or operations shall not be listed unless the profile defines additional requirements for these elements beyond those defined in the schema or in the operations specification (see 7.13.3.3.1).

10.4.7.4.4 Requirements for the specification of property requirements

This subclause details the specification of property requirements in profile specifications, addressing the requirements of 7.13.2.8.

Property requirements not fitting into the "Element requirements" table shall be placed in a separate subclause of the adaptation specific subclause defining the respective adaptation. In this case, the title of the property-specific subclause shall be formatted as follows:

```
PropertySubclauseTitle = "Property" *WSP ":" WS [ AdaptationName *WSP ":"
*WSP ] PropertyName [ "[ ]" ]
```

The square brackets after `PropertyName` are required for array properties.

As required in 7.13.2.8, property requirements should specify a relationship to the aspect of managed objects represented by adaptation instances that is reflected by the property.

Property requirements may specify value constraints (see 7.13.2.8.4); in this case, the conventions defined in 10.2.4 shall be applied.

Property requirements may specify a default value, as detailed in 10.2.4.1.

Property requirements of adaptations with the "instantiated" implementation type may contain input value requirement (see 7.13.2.11); if present, input value requirements shall be specified as defined in 10.4.7.4.5.

Property requirements on CIM references shall state the multiplicity, as detailed in 10.2.4.2.

10.4.7.4.5 Requirements for the specification of input value requirements

Input value requirements may be specified as part of property requirements (see 10.4.7.4.4), or as part of parameter requirements in method requirements (see 10.4.7.4.6).

Requirements for input values defined by the subject profile shall be provided in an input value requirements table.

An input value requirements table shall be labeled:

```
InputValueTableTitle = ElementName "( )" *WSP ":" WS ValueType "value
requirements"

ElementName = PropertyName / ParameterName

ValueType = "Initialization" / "Modification" / "Input"
```

`ElementName` is the name of the property or parameter for which input value requirements are specified. For properties, only the value types "`Initialization`" and "`Modification`" apply; for parameters only the value type "`Input`" applies.

In Table 15, requirements for columns in input value requirements tables are defined. Each required 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 cells under that column.

Table 14 – Requirements for columns in "Input value requirements" tables

Input value – Cell values shall state the required input value.

Requirement – Cell values shall state the requirement level of the input value requirement. The requirement level shall be stated in conformance to the conventions defined in 10.2.1.

Description – Cell values shall provide details about the use of the input value as required by the subject profile.

The following rules apply:

- If the schema descriptions of a specific input value adequately describe its use as required by the subject profile, then the method-specific subclause shall refer to the method parameter description in the schema with the statement "See schema description".
- Unless fitting into a reasonable space within the table cell (about 20 words), the input value requirement description should be placed in a subclause of the method-specific subclause and referenced from the table cell.

10.4.7.4.6 Requirements for the specification of method requirements

This subclause details the specification of method requirements in profile specifications, addressing the requirements of 7.13.3.2, namely the specification of constraints on methods and their parameters according to the requirements of 7.13.3.2.2, the specification of the method semantics as required in 7.13.3.2.3 and the specification of the reporting of method errors as required in 7.13.3.2.4.

Method requirements not fitting into the "Element requirements" table defined in 10.4.7.4.3 shall be placed in a separate subclause of the adaptation specific subclause defining the respective adaptation; this applies to all method requirements that define parameter requirements.

If specified, the title of the method-specific subclause shall be formatted as follows:

```
    MethodSubclauseTitle = "Method" *WSP ":" WS [ AdaptationName *WSP ":" *WSP
    ] MethodName "( )"
```

If stated, `AdaptationName` shall be the name of the adaptation. `MethodName` shall be the name of the method as defined by the profile.

If the method requirement is defined with a requirement level other than "mandatory", the requirement level shall be repeated, applying the conventions defined in 10.2.1.

The method description shall detail the semantics of the method in prose text, addressing the requirements of 7.13.3.2.3. The method description may contain informal references to use cases (see 10.4.9).

Requirements for method parameters defined by the subject profile shall be provided in a method parameter requirements table.

A method parameter requirements table shall be labeled:

```
    MethodParameterTableTitle = [ AdaptationName *WSP ":" WS ] MethodName
    "( )" *WSP ":" WS Parameter requirements"
```

In Table 15, requirements for columns in method parameter requirements tables are defined. Each required 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 cells under that column.

Table 15 – Requirements for columns in "Method parameter requirements" tables

Name – Cell values shall state the parameter name.

Description – Cell values shall provide details about the use of the parameter as required by the subject profile.

The following rules apply:

- If the effective value of one or more of the following qualifiers:

  - In, Out, Required

  defined by the schema definition of the adapted class is True for a method parameter, the name of that qualifier shall be listed first in the description of the method parameter in the method parameter table; if the effective value is False, the name of the qualifier shall not be listed. If more than one of these qualifiers have the effective value True, their names shall be presented in the form of a comma separated list. The convention requirements defined in 10.4.7.4.2 apply.

- If the schema descriptions of a parameter adequately describe its use as required by the subject profile, then the method-specific subclause shall refer to the method parameter description in the schema with the statement "See schema description".

- Value constraints may be specified; in this case, the conventions defined in 10.2.4 shall be applied.

- – A default value may be specified, as detailed in 7.13.2.9
- – Unless fitting into a reasonable space within the table cell (about 20 words), the description should be placed in a subclause of the method-specific subclause that is referenced from the table cell.
- – If input parameter value requirements (see 7.13.2.11.4) are specified for a parameter, then the parameter description shall be placed in a subclause of the method-specific subclause that is referenced from the "Description" table cell. In this case the parameter specific subclause shall also contain the input parameter value requirements, in the format required in 10.4.7.4.5.

NOTE Version 1.0 of this guide defined a Qualifiers column and a Type column; these were dropped with version 1.1 of this guide. Instead, the requirement for repeating the effective value of schema defined qualifiers was replaced by the first rule defined for the Description column above; repeating the schema defined type of a parameter is no longer required. The former "Description/Values" column is now titled "Description" for coherent definition of tables specified in this guide.

The method parameter requirements table shall contain a special parameter named "`ReturnValue`" that describes the use of return values as required by the subject profile.

If the schema definition of method return values does not adequately describe their use as required by the subject profile, that description shall be provided in the corresponding cell in the method parameter requirements table or a subclause referenced from there.

If the schema definition of method return values adequately describe their use as required by the subject profile, the description should refer to the schema. For example, an Example 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."

The reporting of method errors as required in 7.13.3.2.4 shall be specified as follows:

- If the subject profile defines requirements for standard messages for a method, these shall be stated as defined in 10.4.7.4.9.
- If the subject profile defines additional constraints on CIM status codes for a method, these shall be stated as defined in 10.4.7.4.9.

10.4.7.4.7 Requirements for the specification of operation requirements

Operation requirements not fitting into the "Element requirements" table shall be placed in a separate subclause of the adaptation specific subclause defining the respective adaptation. In this case, the title of the operation-specific subclause shall be formatted as follows:

```
OperationSubclauseTitle = "Operation" *WSP ":" WS [ AdaptationName *WSP
":" *WSP ] OperationName "( )"
```

If stated, `AdaptationName` shall be the name of the adaptation. `OperationName` shall identify the operation (that is defined in the operations specification - see 7.13.3.3.1) for that operation requirements are defined; see 10.4.7.4.2. The operation requirements shall be based on the definition of operations in the operations specification.

If the operation requirement is defined with a requirement level other than "mandatory", the requirement level shall be repeated, applying the conventions defined in 10.2.1.

Operation requirements may extend the behavior defined in the referenced operations specification (for example, by requiring specific effects on the managed environment); the description of such extensions should include all side effects and expected results in the managed environment.

The reporting of operation errors as required in 7.13.3.3.6 shall be specified as follows:

- If the subject profile defines requirements for standard messages for an operation, these shall be stated as defined in 10.4.7.4.9.

- If the subject profile defines additional constraints on CIM status code values for an operation, these shall be stated as defined in 10.4.7.4.9.

10.4.7.4.8 Requirements for the specification of operations related to association traversal

Operations that result in associated or association instances (or instance paths) relative to a source instance are called association traversal operations. Profiles shall define the requirements for association traversal operations as part of the operation requirements of adaptations that are referenced by association adaptations, not as part of the operation requirements of the association adaptations themselves.

In addition, a particular adaptation defined by the subject profile can be the source point for the traversal of more than one association adaptation. If in this case the requirements are different for each association adaptation that can be traversed, then separate operation requirements are required for each traversable association within the definition of that source adaptation.

For example, if a profile defines operations as defined in DSP0223 in order to traverse its SystemDevice adaptation of the CIM_SystemDevice association, the requirements for association traversal operations such as the GetAssociatedInstances( ) and GetAssociatedInstancePaths( ) operations would not be specified as part of the operation requirements of the SystemDevice adaptation; instead, the operation requirements for association traversal operations would be specified as part of the operation requirements of adaptations referenced by the SystemDevice association adaptation, in this case for example a System adaptation of the CIM_System class and a LogicalDevice adaptation the CIM_LogicalDevice class.

NOTE Associations may be adapted such that adaptations of subclasses of the classes referenced by the adapted association are referenced; see 7.13.2.8.

EXPERIMENTAL

10.4.7.4.9 Requirements for the specification of error reporting requirements

If the subject profile does not define error reporting requirements for a method (see 7.13.3.2.4) or operation (see 7.13.3.3.6), no error reporting requirements shall be defined in the method-specific or operation-specific subclause; instead, the subclause should contain a statement such as "No error reporting requirements are defined." Alternatively, if the operations specification (see 7.13.3.3.1 and 10.4.7.4.2) defines error reporting requirements, a statement such as

```
"For error reporting requirements, see" OpSpec "."
```

should be used, with `OpSpec` referring to the operations specification.

NOTE These statements are not required for method or operation requirements solely described through a table entry in the "Element requirements" table (see 10.4.7.4.3), because in this case there is no method-specific or operation-specific subclause.

If a profile defines error reporting requirements (see 7.13.3.2.4 and 7.13.3.3.6), these shall be defined in an error reporting requirements table.

The error reporting requirements table shall be labeled as follows:

```
ErrorReportingRequirementsTableTitle = ActivityName "( )" *WSP ":" WS
Error reporting requirements"
```

```
ActivityName = MethodName / OperationName
```

`MethodName` is name of the method defined in the profile for which error reporting requirements are defined. `OperationName` is name of the operation (defined in the operations specification - see 7.13.3.3.1) for which the profile defines profile-specific error reporting requirements.

In Table 16 requirements for columns of the error reporting requirements table are defined. Each column is described by an entry in the list provided in Table 16. 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 16 – Requirements for columns of the "Error reporting requirements" table

Reporting mechanism – Each cell values shall identify an error reporting mechanisms.

The following rules apply:

– Error reporting mechanisms shall be listed using the following format:

```
ErrorReportingMechanism = MessageIdentificationList / CimStatusCode

MessageIdentificationList = MessageIdentification [ WS "," WS
    MessageIdentificationList ]

MessageIdentification = MessageRegistryRefName "::" MessageID
```

– `MessageRegistryRefName` shall be the message registry reference name (see 10.4.5.7) of the registry in which the standard error message is defined, and `MessageID` shall be the message id of that error message. The message id is the concatenation of the value of the PREFIX attribute and the SEQUENCE_NUMBER attribute from the MESSAGE_ID element that describes the message in the message registry.

`CimStatusCode` shall be a CIM status code.

– The order of error reporting mechanisms listed in the table does not establish an order for their selection in case of respective error situations. However, a profile may establish that interpretation for individual or for all error reporting requirements specified in the profile. Note that some operations specifications imply an order for in their error reporting requirements.

Requirement – Cell values shall state the requirement level of the input value requirement.

The requirement level shall be stated in conformance to the conventions defined in 10.2.1.

Description – Cell values shall state the message text (abbreviated, if appropriate).

– Unless fitting into a reasonable space within the table cell (about 20 words), the message description should be placed in a separate subclause and referenced from the table

An example of an error reporting requirements table is provided in A.4.4.

EXPERIMENTAL

DEPRECATED

Minor revisions of profiles written in conformance with version 1.0 of this guide may continue using a format as defined by Table 17 instead of the format defined in Table 16. However, return values and messages are alternatives. Profiles should not define the use of return values for situations that result in a CIM error, because in this case the method or operation does not return and no return value is returned. Either an operation or method is successful at the operations level and returns a return value, or it is not successful at the operations level, resulting in a CIM error containing zero or more messages.

Table 17 – 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 a reasonable amount of words (about 20 words). If more text is required, respective content shall be placed in a separate subclause and referenced from the table.

DEPRECATED

10.4.7.4.10 Requirements for the specification of metric requirements

Metric requirements not fitting into the table defined in 10.4.7.4.3 shall be placed in a separate subclause of the subclause defining the respective adaptation.

If specified, the title of the metric-specific subclause shall be formatted as follows:

```
    MetricSubclauseTitle = "Metric: " MetricName
```

`MetricName` shall be the name of the metric as defined in the referenced metric registry.

If the metric requirement is defined with a requirement level other than "mandatory", the requirement level shall be repeated, applying the conventions defined in 10.2.1.

Metric requirements should detail the semantics of the metric as required in 7.13.3.5.

10.4.7.4.11 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 7.13.3.4. Instance requirements may be specified as part of the implementation requirements, or may be specified in a separate subclause.

10.4.7.4.12 Requirements for the specification of indication-generation requirements

Each adaptation definition subclause that defines an adaptation of an indication class shall state indication-generation requirements, as defined in 7.13.4.1. Indication-generation requirements may be specified as part of the implementation requirements, or may be specified in a separate subclause.

DEPRECATED

Profile specifications that apply the condensed profile specification structure (see 10.3.2) shall not contain a "Methods" clause because in this case respective content is already specified as part of adaptation definitions within the "Implementation" clause; see 10.4.7.4.6 and 10.4.7.4.7.

10.4.8 Requirements for the specification of the "Methods" clause

This subclause details requirements for the "Methods" clause in profile specifications.

10.4.8.1 General

Profile specifications that apply the traditional profile specification structure (see 10.3.3) shall contain a "Methods" clause.

10.4.8.2 Requirements for the specification of methods

This subclause specifies the definition of method requirements in profile specifications that apply the traditional profile specification structure.

10.4.8.2.1 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 contain only 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 contain only the statement "No method requirements are defined."

10.4.8.2.2 Method-specific subclauses

Each extrinsic method that is referenced by a class adaptation defined in a 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 10.4.10.3.

The method-specific subclause should provide a description detailing the semantics of the method as required in 7.13.3.2. The description may contain references to use cases (see 10.4.9).

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 18 requirements for columns in method parameter tables are defined. Each required column is described by an entry in the list provided in Table 18. 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 18 – 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; it shall 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; however, a profile may establish value constraints on parameters (see 7.13.3.2).
- 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.

The following rules apply:

- If value constraints are defined, the conventions defined in 10.2.4 shall be applied.
- The value in a Description/Value table cell should contain not more than a reasonable amount of words (about 20 words). Longer text passages should be placed in a subclause of the method-specific subclause and referenced from the table cell.

If the schema descriptions of method parameters adequately describe the use of the method parameters as required by the subject profile, then the method-specific subclause shall refer to the method parameter description in the schema with this statement: "See schema description."

If the schema descriptions of method return values does not adequately describe their use as required by the subject profile, the method-specific subclause shall provide a table specifying return values.

The table shall be labeled:

```
ReturnValueTableTitle = MethodName "( ): Return values"
```

In Table 19 requirements for columns of the return value table are defined. Each column is described by an entry in the list provided in Table 19. 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 19 – Requirements for columns of the return value table

Value – Cell values shall state the numeric return value followed by the corresponding string description in parentheses. The description shall not be enclosed in quotes.

Example: `"1 (Not Implemented)"`.

Description – Cell values shall provide details about the situation indicated by the return value.

The following rules apply:

- If a return value only applies under certain conditions, this shall be stated in the following form:

  ```
  "Applicable only if the " ConditionalElement " is implemented."
  ```

- The value in a Description table cell should contain not more than a reasonable amount of words (about 20 words). Longer text passages 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, an Example 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 10.4.7.4.9. 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."

10.4.8.3 Requirements for the specification of the "Operations" subclause

This subclause details requirements for the "Operations" subclause of the "Methods" clause in profile specifications.

10.4.8.3.1 General

The "Methods" clause should 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 contain only the statement: "All operation requirements are defined in base profile(s)."

10.4.8.3.2 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 7.13.3.3. For example, "This profile defines operations in terms of DSP0223."

Table 20 defines three options, one of which shall be applied by a profile specification for the "Generic operations" subclause.

Table 20 – Profile convention options

Option	Requirements for the Intrinsic operations subclause
Option 1 – Table includes each operation for each class.	**Deprecated** with version 1.0.1; replaced by option 2, with additional requirements specified in 10.4.8.3.3. "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 profile class (including associations), the implementation requirements for operations, including for those in the following default list, are specified in class-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 applicability of the default list shall be specified in adaptation-specific subclauses of the "Operations" subclause of the "Methods" clause; see 10.4.8.3.3.
Option 3 – Table includes operations with profile-specific requirements. Other operations may be implemented.	**Deprecated** with version 1.0.1; replaced by option 2, with additional requirements specified in 10.4.8.3.3. "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.

10.4.8.3.3 Requirements for the specification of class-specific operations subclauses

A subclause shall be included for each class adaptation (including association adaptations) defined by the subject profile.

Subsequent definitions in this subclause make use of the following ABNF rules:

- `TableNum` is the number of the table.
- `OpSpec` is a reference to the operations specification.
- `PcoNum` is the subclause number of the "Profile conventions for operations" subclause.

If a default list of operations was specified, and the profile does not require modifications on that default list, 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   Related profiles may define additional requirements on operations for the
        profile class."
```

If a default list of operations was specified, and the profile requires modifications on that default list, the modification shall be stated in a separate table, and the following statement (including the NOTE) shall be provided:

```
    "Table " TabNum " lists implementation requirements for operations. If
    implemented, these operations shall be implemented as defined in " OpSpec
    ". In addition, and unless otherwise stated in Table " TabNum ", all
    operations in the default list in " PCONum " shall be implemented as
    defined in " OpSpec "."

    "NOTE   Related profiles may define additional requirements on operations for the
            profile class."
```

NOTE The quotation, the indentation and the use of a monospaced font are elements of the ABNF rule and are not part of the normative definition. Instead, the presented text is intended to be part of the normal text of the subject profile.

If a table is provided detailing requirements for operations, the table shall have the format as defined in 10.4.7.4.7.

For operations related to associations the requirements defined in 10.4.7.4.8 apply correspondingly for "profile classes".

DEPRECATED

10.4.9 Requirements for the specification of the "Use cases" clause

This subclause details requirements for the "Use cases" clause in profile specifications.

10.4.9.1 General

Each profile specification shall have a "Use cases" clause.

Within the "Use cases" clause, each use case defined by the profile (see 7.16) shall be documented in a separate subclause, as detailed in 10.4.9.3.

State descriptions (see 7.16.2) may be documented as part of a use case, or may be documented in a separate subclause of a "Use cases" clause that is referenced from within use case specific subclauses.

10.4.9.2 Requirements for the specification of subclauses containing state descriptions

A profile specification may contain zero or more subclauses with state descriptions depicting typical situations that a client may observe in the process of applying use cases defined by the profile. Each state description-specific subclause shall contain one state description.

All or part of a state description may be provided in graphical form as DMTF object diagrams; in this case, the rules defined in 8.3.7 apply.

The title of state description subclauses shall be formatted as follows:

```
    StateDescriptionSubclauseTitle = [ "StateDescription *WSP ":" *WSP ]
    StateDescriptionName [ *WSP ":" *WSP StateDescriptionTitle ]
```

`StateDescriptionName` shall state the name of the state description. The name shall comply with the rules for names of named profile elements (see 7.2.2), and should be chosen such that it enables a human reader to grasp the situation detailed by the state description; the name shall be unique within the profile specification. `StateDescriptionTitle` may state a phrase that further details the purpose of the state description in situations where `StateDescriptionName` does not suffice.

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.

10.4.9.3 Requirements for the specification of use-case-specific subclauses

10.4.9.3.1 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 [ *WSP ":" *WSP UseCaseTitle ]
```

`UseCaseName` shall state a name for the use case. The name shall comply with the rules for names of named profile elements (see 7.2.2), and should be chosen such that it enables a human reader to grasp the intent of the use case; the name shall be unique within the profile. `UseCaseTitle` may state a phrase that captures the purpose of the use case in situations where `UseCaseName` does not suffice.

Each use case-specific subclause should contain a brief description of the use case.

See A.5 for examples of use cases.

10.4.9.3.2 Requirements for the specification of preconditions in use cases

The definition of preconditions as required by 7.16.3 shall be provided within a first subclause within any the use case-specific subclause. The precondition subclause shall be titled "Preconditions".

Sequences of statements expressing elements of preconditions should be organized in a list format.

10.4.9.3.3 Requirements for the specification of flows of activities in use cases

The description of flows of activities as required by 7.16.4 shall be provided in a separate subclause within any use case-specific subclause. The subclause shall be titled "Flow of activities".

The following formal requirements apply:

- Use case steps should be numbered. Numbering is required if use case steps are referenced.
- Descriptions may contain references to DMTF object diagrams.
- Normative requirements shall not be duplicated in use case descriptions.
- Parameter values should be stated in a list format where each list entry describes one parameter and its value. If a parameter value 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 flow of activities should be the sequential processing of use case steps; however, the following phrases may be used to indicate special situations:
  – `StepPostCondition "; the use case continues with step" StepNumber "."`

where `StepPostCondition` 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 step should be listed together with the respective post condition.

– `"This completes the use case; the postconditions in" SubclauseNumber "apply."`

This phrase describes a normal completion of the use case. Within the description of one use case at least one step should end with a normal completion of the use case.

– `"This terminates the use case; the postconditions in" SubclauseNumber "apply."`

This phrase describes an abnormal termination of the use case. Within the description of one use case zero or more steps can end with an abnormal termination of the use case.

Alternatively to the format defined above, use cases may be presented as pseudo-code.

10.4.9.3.4 Requirements for the specification of postconditions in use cases

The definition of a postcondition as required by 7.16.5 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 situations resulting from processing the use case such as success or failure. Such situations may likewise be presented by other structuring elements such as lists; however, separate subclauses are required if the content is referenced elsewhere.

DEPRECATED

Profile specifications that apply the condensed profile specification structure (see 10.3.2) shall not contain a "CIM elements" clause because in this case the definition of CIM elements is replaced by the definition of class adaptations within the "Implementation" clause (see 10.4.7.4), and the list of class adaptations is provided as part of the "Synopsis" clause (see 10.4.5).

10.4.10 Requirements for the specification of the "CIM elements" clause

This subclause details requirements for the "CIM elements" clause in profile specifications.

10.4.10.1 General

Each profile specification that applies the traditional profile specification structure (see 10.3.3) shall contain a "CIM elements" clause.

Version 1.0 of this guide did not formally define the concept of adaptations; instead it informally used the terms "class", "profile class", "profiled class", or "supported class". For details, see 7.13.1.

Revisions of existing profile specifications that apply version 1.1 or a later version of this guide 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 profiles written conformant to 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 10.4.10.2.
- A subclause for each adaptation defined by the profile; see 10.4.10.3.

10.4.10.2 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 association adaptations and indication adaptations). The table shall be labeled:

```
CIMElementTableTitle = ProfileName "profile : CIM elements"
```

`ProfileName` shall be the registered name of the profile. Each entry in the table shall declare an adaptation defined by the subject profile.

The table shall have four columns:

AdaptationName – Cell values shall state the name of the adaptation; see 7.13.

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 7.6.2) 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 message shall be stated in a subsequent subcell.

Requirement – Cell values shall state the requirement level for the adaptation, as defined in 10.2.1.

The following rules apply:

– 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 10.2.3.

– 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, in which 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 20 words, a separate adaptation definition subclause shall be provided within the "Implementation" clause; for details, see 10.4.7.4.3. 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.

10.4.10.3 Requirements for the specification of subclauses defining class adaptations

The specification of the each class adaptation subclause shall be in compliance with 10.4.7.4, with the following admissible deviations:

- The title of the subclause may apply the deprecated naming convention using the name of the adapted class and a modifier; for details see 7.13.

DEPRECATED

Annex A
(Informative)

Examples

A.1 General

All the examples provided within Annex A provide excerpts from a hypothetical Example Fan profile. The examples are related to each other, but together they would not form a complete profile specification.

A.2 Example of a "Synopsis" clause

Table A.1 provides an example of a "Synopsis" clause; see 10.4.5 for requirements on the specification of the "Synopsis" clause.

Table A.1 – Example of "Synopsis" clause

X-5 Synopsis

X-5.1 Profile attributes

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

X-5.2 Summary

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.

X-5.3 Profile references

Table X-1 lists the profile references defined in this profile.

Table X-1 – Profile references

Profile reference name	Profile name	Organi-zation	Version	Relationship	Description
Indications	Indications	DMTF	1.2	Conditional	The profile defining the creation and delivery of indications. Condition: The Indications feature is implemented; see **X-7.2.1** for feature definition.
FanProfileRegistration	Example Profile Registration	DMTF	1.1	Mandatory	The Example Profile Registration profile applied for the registration of implementations of the Example Fan profile.
FanPhysicalAsset	Example Physical Asset	DMTF	1.1	Optional	The Example Physical Asset profile applied for fans as physical assets.
FanSensors	Example Sensors	DMTF	1.1	Conditional	The Example Sensors profile applied for sensors of fans. Condition: The FanSpeedSensor feature is implemented; see X-7.2.4 for the feature definition.

X-5.4 Referenced registries

Table X-2 lists the message registry references defined by this profile.

Table X-2 – Message registry references

Registry reference name	Registry name	Organization	Version	Description
WBEMMREG	WBEM Operations Message Registry	DMTF	1.0	See DSP8016.
PLATMREG	Platform Alert Message Registry	DMTF	1.1	See DSP8007.

X-5.5 Features

Table X-3 lists the features defined in this profile.

Table X-3 – Features

Feature name	Granularity	Requirement	Description
Indications	Profile	Optional	See **X-7.2.1** for feature definition.
FanStateManagement	Fan instance	Optional	See X-7.2.2 for feature definition.

FanElementNameModification	Fan instance	Optional	(Not detailed in this example)
FanSpeedSensor	Fan instance	Conditional	See X-7.2.4 for feature definition.
FanLifecycleAlerts	Profile	Conditional	See X-7.2.5 for feature definition.

X-5.7 Adaptations

Table X-4 lists the class adaptations defined in this profile.

Table X-4 – Adaptations

Adaptation	Elements	Requirement	Description
Instantiated, embedded and abstract adaptations			
Fan	CIM_Fan	Mandatory	See X-7.4.3.
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.
FanNumericSensor	CIM_NumericSensor	Conditional	See X-7.4.8.
SensorOfFan	CIM_AssociatedSensor	Conditional	See X-7.4.9.
…	…	…	…
FanProfileRegistration	CIM_RegisteredProfile	Mandatory	See …
…	…	…	…
FanSystem	CIM_System	Mandatory	Instantiated ordinary adaptation; scoping class adaptation; scoping profiles base their central class adaptation on this adaptation.
…	…	…	…
Indications and exceptions			
FanAddedAlert	CIM_AlertIndication	Conditional	See X-7.4.34.
FanRemovedAlert	CIM_AlertIndication	Conditional	See X-7.4.35.
FanFailedAlert	CIM_AlertIndication	Optional	See X-7.4.36.
FanReturned-ToOKAlert	CIM_AlertIndication	Optional	See X-7.4.37.
FanDegradedAlert	CIM_AlertIndication	Optional	See X-7.4.38.

X-5.8 Use cases

Table X-6 lists the use cases defined in this profile.

Table X-6 – Use cases

Use-case name	Description
…	…
DetermineFanState	See X-8.3.
…	...
RequestFanStateChange	See X-8.7.
…	…

A.3 Example of a "Description" clause

Table A.2 shows an example of the "Description" clause for an Example Fan profile.

Table A.2 – Example of a "Description" clause

X-6 Description

X-6.1 General

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

X-6.1 Fan

A fan is a device within a system that provides active cooling to specific elements of a system, and/or to the system as a whole.

For the management domain addressed by this profile, a fan is considered to be either active or inactive; any other potentially possible state needs to be mappable.

X-6.2 System

A system is an entity made up of components that operates as a 'functional whole'. A system can contain elements that require cooling, such as processors, chipsets, disks or power supplies. Each of these elements may require cooling by means of dedicated fans, and/or may depend on cooling provided to the system as a whole.

X-6.3 Cooled element

Cooled elements are elements contained by a system that require cooling.

X-6.4 Temperature sensor

A temperate sensor measures either the temperature of the system as a whole, or that of individual cooled elements within a system.

X-6.5 Fan speed sensors

Fans speed sensors allow monitoring the rotation speed of fans.

…

X-6.10 CIM model overview

Figure <Fig1> represents the DMTF collaboration structure diagram the Example Fan profile.

NOTE Here one or more DMTF collaboration diagrams and/or DMTF adaptation diagrams would be placed. For examples, see Figure 8 on page 76.

The FanSystem adaptation (see X-6.2) models systems (see X-6.2).

The Fan adaptation (see X-7.4.3) models fans (see X-6.1).

…

A.4 Example of an "Implementation" clause

A.4.1 Example of the general layout of an "Implementation" clause

Table A.3 shows an example of the general layout of the "Implementation" clause; see 10.4.7 for requirements on the specification of the "Implementation" clause.

Table A.3 – 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-7.4 Adaptations

// See A.4.3 for an example of the "General requirements" subclause.

// See A.4.4 for examples of subclauses defining adaptations of ordinary classes and associations.

…

A.4.2 Example of feature definitions

Table A.4 shows examples of feature definitions within the "Features" subclause of the "Implementation" subclause; see 7.15 for requirements on the specification of features.

Table A.4 – Example definitions of features

X-7.2.1 Feature: Indications

X-7.2.1.1 General

The implementation of the Indications feature is conditional.

Condition: Any of the following is true:

- The FanLifecycleAlertsFeature is implemented; see **X-7.2.5**.

- The FanFailedAlert indication adaptation is implemented; see **X-7.4.36**.
- The FanReturnedToOK indication adaptation is implemented; see **X-7.4.37**.
- The FanFailedAlert indication adaptation is implemented; see **X-7.4.38**.

X-7.2.1.2 Feature description

The implementation of the Indications feature provides for indications being generated and delivered to subscribed listeners as the events modeled by these indications occur.

X-7.2.1.3 Feature discovery

The presence of the Indications feature is indicated by the exposure of an Indications::IndicationsProfileRegistration instance (see DSP1054) that is related to the FanProfileRegistration instance (see …) with a ReferencedProfile association instance (see …).

X-7.2.2 Feature: FanStateManagement

X-7.2.1.1 General

The implementation of the FanStateManagement feature is conditional.

Condition: The managed environment includes fans that are state manageable.

X-7.2.1.2 Feature description

The implementation of the FanStateManagement feature enables clients to request state changes on fans, such as activation or deactivation.

X-7.2.1.3 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-7.2.3 Feature: FanElementNameEdit

[not detailed in this example]

…

X-7.2.4 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-7.2.5 Feature: FanLifecycleAlerts

The implementation of the FanLifecycleAlerts feature is optional.

The FanLifecycleAlerts feature groups the requirements for reporting fan lifecycle events such as the addition of a fan to the managed environment, or the removal of a fan from the managed environment.

A.4.3 Example of the "Conventions" subclause

Table A.5 details an example of the "Conventions" subclause within the "Adaptations" subclause of the "Implementation" clause; see 10.4.7.4.2 for requirements on the specification of implementation requirements for operations.

Table A.5 – Example of the "Conventions" subclause

X-7.4.1 Conventions

…

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

- In: indicates that the parameter is an input parameter
- Out: indicates that the parameter is an output parameter
- Key: indicates that the property is a key (that is, its value is part of the instance part)
- Required: indicates that the element value shall be non-Null.

This profile defines operation requirements based on DSP0223.

> For adaptations of ordinary classes and of associations the requirements for operations are specified in adaptation-specific subclauses of X-7.4.
>
> For association traversal operation requirements that are specified only in the elements table of an adaptation (i.e. without operation-specific subclauses), the names of the association adaptations to be traversed are listed in the elements table.
>
> …

A.4.4 Examples of subclauses defining adaptations

Table A.6 details examples of subclauses within the "Adaptation" subclause of the "Implementation" clause that define adaptations of ordinary classes and associations; see 10.4.7.4 for requirements on the specification of class adaptations.

Table A.6 – Examples of subclauses defining adaptations

X-7.4.3 Fan: CIM_Fan

X-7.4.3.1 General

The Fan adaptation models fans in systems; fans are described in X-6.1.

The implementation type of the Fan adaptation is: "instantiated".

The Fan adaptation shall conform to the requirements for central elements as defined by the Profile Registration profile (see DSP1033).

Table X8 lists the element requirements of the Fan adaptation.

Table X8 – Fan: Element requirements

Element	Requirement	Description
Base adaptations		
ExampleSensors::SensoredElement	Conditional	Condition: The FanSpeedSensor feature is implemented; see X-7.2.4. See DSPxxxx.
Properties		
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-7.2.4. See CIM schema definition.
ActiveCooling	Mandatory	Value shall be True
EnabledState	Mandatory	See X-7.4.3.3.
RequestedState	Conditional	Condition: The FanStateManagement feature is implemented; see X-7.2.2. See X-7.4.3.4.
ElementName	Conditional	Condition: The FanElementNameManagement feature is implemented; see X-7.2.3. See CIM schema definition.
Methods		
RequestStateChange( )	Conditional	Condition: The FanStateManagement feature is implemented; see X-7.2.2. See X-7.4.3.5.
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.

GetClassInstancePaths( )	Mandatory	See DSP0223.
GetAssociatedInstancesWithPath( )	Mandatory	See DSP0223.
GetAssociatedInstancePaths( )	Mandatory	See DSP0223.
GetReferencingInstancesWithPath( )	Mandatory	See DSP0223.
GetReferencingInstancePaths( )	Mandatory	See DSP0223.
ModifyInstance( )	Optional	See X-7.4.3.6, and DSP0223.

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( )

X-7.4.3.4.1 General

The requirement level of the RequestStateChange( ) method is conditional.

Condition: The FanStateManagement feature is implemented; see X-7.2.2.

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 requested state.
- If the requested state is 2 (Enabled), the implementation shall execute an activation of the target fan.
- If the requested state is 3 (Disabled), the implementation shall execute a deactivation of the target fan.
- Any other requested state shall be rejected, issuing messages WBEMMREG::WIPG0227 and PLATMREG::PLATxxx1.
- Depending on the outcome of the operation executed by the implementation, the resulting state shall be reflected by the value of the EnabledState property.

Table X-9 lists the parameter requirements for the RequestStateChange( ) method.

Table X-9 – RequestStateChange( ): Parameter requirements

Name	Description
RequestedState	In, see X-7.4.3.4.2.
TimeoutPeriod	In, see X-7.4.3.4.3.
Job	Out, see X-7.4.3.4.4.
ReturnValue	See schema definition.

X-7.4.3.4.2 RequestedState

A non-Null instance path shall be returned if a job was started; otherwise, Null shall be returned.

X-7.4.3.4.3 TimeoutPeriod

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

- 0 or Null – No maximum time is specified
- Non-Null – The value specifies the maximum time allowed

Note that for the case that the value is Non-Null and not 0, and the implementation is unable to support the semantics of the TimeoutPeriod parameter, the schema definition of the adapted class requires that the value 4098 (Use of Timeout Parameter Not Supported) is returned.

X-7.4.3.4.4 Job

A ConcreteJob (see …) instance path shall be returned if a job was started; otherwise, Null shall be returned.

X-7.4.3.4.6 Error reporting requirements

Table X-11 specifies the error reporting requirements for the RequestStateChange( ) method. These requirements apply on top of those required by DSP0223 for the InvokeMethod( ) operation.

Table X-11 – RequestStateChange( ): Error reporting requirements

Reporting mechanism	Requirement level	Description
WBEMMREG::WIPG0208, PLATMREG::PLAT9001	Mandatory	The requested state is not supported for the fan.
WBEMMREG::WIPG0208, PLATMREG::PLAT9002	Mandatory	A non-Null value for the Timeout parameter is not supported.
WBEMMREG::WIPG02019	Mandatory	Method is not implemented.
WBEMMREG::WIPG0227, PLATMREG::PLAT9003	Mandatory	Fan cannot be disabled due to excessive temperature. The detail text of WIPG0227 should be omitted or should indicate that the next message details the error.

WBEMMREG::WIPG0227	Mandatory	Any other failure. As defined in WIPG0227, the failure shall be described in its detail text.
CIM_ERR_SERVER_LIMITS_EXCEEDED	Mandatory	More element changes are under way than the configured limit of concurrent changes, or there is a resource shortage in the WBEM server.

…

X-7.4.3.5 Operation: ModifyInstance( )

The implementation of the ModifyInstance( ) operation for the Fan adaptation is optional.

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 which the modification is requested; this instance is referred to as the *target instance* in this subclause. All other properties in the input instance shall be ignored. The fan in the managed environment that is represented by the target instance is referred to as the *target fan* in this subclause. Using these terms, 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.

Table X-12 specifies the error reporting requirements of the ModifyInstance( ) method. These requirements apply on top of those required by DSP0223 for the ModifyInstance( ) operation.

Table X-12 – ModifyInstance( ): Error reporting requirements

Reporting mechanism	Requirement level	Description
WBEMMREG::WIPG0227, PLATMREG::PLATxxx1	Mandatory	Operation not supported for the fan
WBEMMREG::WIPG0227, PLATMREG::PLATxxx2	Mandatory	Temperature too high for disabling the fan
WBEMMREG::WIPG0227, PLATMREG::PLATxxx3	Mandatory	Insufficient power for enabling the fan

…

X-7.4.4 Adaptation: FanInSystem: CIM_SystemDevice

The FanInSystem association adaptation models the relationship between fans and their containing system.

The implementation type of the FanInSystem adaptation is: "instantiated".

Each Fan (see X-7.4.3) instance shall be associated through a FanInSystem instance to the FanSystem (see …) instance representing the system containing the fan.

Table X-13 lists the implementation requirements for the FanInSystem adaptation.

Table X-13 – FanInSystem: Element requirements

Element	Requirement	Description
Properties		
GroupComponent	Mandatory	**Key**: Value shall reference the System instance representing the system that contains the fan **Multiplicity**: 1
PartComponent	Mandatory	**Key**: Value shall reference the Fan instance representing a fan **Multiplicity**: *
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.

X-7.4.5 Adaptation: FanCapabilities: CIM_EnabledLogicalElementCapabilities

The FanCapabilities adaptation models the capabilities of fans in managed systems.

The requirement level of the FanCapabilities adaptation is conditional.

Condition: One or more of the following conditions:

- The FanStateManagement feature is implemented; for feature definition see X-7.2.2.
- The FanElementNameEdit feature is implemented; for feature definition see X-7.2.3.

The implementation type of the FanCapabilities adaptation is: "instantiated".

For each fan supporting the FanStateManagement feature or the FanElementNameEdit feature the capabilities of that fan shall be represented by a FanCapabilities instance.

Table X-14 lists the element requirements for this class adaptation.

Table X-14 – FanCapabilities: Element requirements

Element	Requirement	Description
Properties		
RequestedStatesSupported[ ]	Conditional	Condition: The FanStateManagement feature is implemented; see X-7.2.2. See CIM schema definition.
ElementNameEditSupported	Conditional	Condition: The ElementNameEdit feature is implemented; see X-7.2.3. If the ElementNameEdit feature is supported, the value shall be True,

		otherwise False.
MaxElementNameLen	Conditional	Condition: The ElementNameEditSupported property is implemented. See CIM schema definition.
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.
GetAssociatedInstancesWithPath( )	Mandatory	See DSP0223.
GetAssociatedInstancePaths( )	Mandatory	See DSP0223.
GetReferencingInstancesWithPath( )	Mandatory	See DSP0223.
GetReferencingInstancePaths( )	Mandatory	See DSP0223.

X-7.4.6 Adaptation: CapabilitiesOfFan: CIM_ElementCapabilities

The CapabilitiesOfFan adaptation models the relationship between a fan and its capabilities.

The requirement level of the CapabilitiesOfFan adaptation is conditional.

Condition: The FanCapabilities adaptation is implemented; see X-7.4.5.

The implementation type of the CapabilitiesOfFan adaptation is: "instantiated".

Each FanCapabilities (see X-7.4.5) instance shall be associated through a CapabilitiesOfFan instance to the Fan (see X-7.4.3) instance for which it represents capabilities.

Table X-15 lists the element requirements for this association adaptation.

Table X-15 – CapabilitiesOfFan: Element requirements

Element	Requirement	Description
Properties		
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
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.

X-7.4.7 Adaptation: FanSensor: CIM_Sensor

The FanSensor adaptation models fans with discrete speed sensors.

The requirement level of the FanSensor adaptation is conditional.

Condition: All of the following:

- The FanSpeedSensor feature is implemented (see X-7.2.4).
- Fan speed sensors within the managed environment support reporting discrete speed.

The implementation type of the FanSensor adaptation is: "instantiated".

Fan speed sensors within the managed environment that support reporting discrete speed may be represented by FanSensor instances.

Table X-16 lists the element requirements for this class adaptation.

Table X-16 – FanSensor: Element requirements

Element	Requirement	Description
Base adaptations		
FanSensors::Sensor	Mandatory	See DSPxxxx.
Properties		
SensorType	Mandatory	Value shall be 5 (Tachometer).
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.
GetAssociatedInstancesWithPath( )	Mandatory	See DSP0223.
GetAssociatedInstancePaths( )	Mandatory	See DSP0223.
GetReferencingInstancesWithPath( )	Mandatory	See DSP0223.
GetReferencingInstancePaths( )	Mandatory	See DSP0223.

X-7.4.8 Adaptation: FanNumericSensor: CIM_NumericSensor

The FanNumericSensor adaptation models fan speed sensors that report analogous speed.

The requirement level of the FanNumericSensor adaptation is conditional.

Condition: All of the following:

- The FanSpeedSensor feature is implemented; see X-7.2.4.
- Fan speed sensors within the managed environment support reporting analogous speed.

The implementation type of the FanNumericSensor adaptation is: "instantiated".

Table X-17 lists the element requirements for this class adaptation.

Table X-17 – FanNumericSensor: Element requirements

Elements	Requirement	Notes
Base adaptations		
FanSensors::NumericSensor	Mandatory	See DSPxxxx.
Properties		
SensorType	Mandatory	Value shall be 5 (Tachometer)
BaseUnits	Mandatory	Value shall be 19 (RPM)
RateUnits	Mandatory	Value shall be 0 (None)
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.
GetAssociatedInstancesWithPath( )	Mandatory	See DSP0223.
GetAssociatedInstancePaths( )	Mandatory	See DSP0223.
GetReferencingInstancesWithPath( )	Mandatory	See DSP0223.
GetReferencingInstancePaths( )	Mandatory	See DSP0223.

X-7.4.9 Adaptation: SensorOfFan: CIM_AssociatedSensor

The SensorOfFan adaptation models the relationship between fans and their sensors.

The requirement level of the SensorOfFan adaptation is conditional.

Condition: The FanSpeedSensor feature is implemented; for feature definition see X-7.2.4**.**

The implementation type of the SensorOfFan adaptation is: "instantiated".

Each FanSensor (see X-7.4.7) or FanNumericSensor (see X-7.4.8) instance shall be associated through a SensorOfFan instance to the Fan instance representing the monitored fan.

Table X-18 lists the element requirements for this association adaptation.

Table X-18 – SensorOfFan: Element requirements

Element	Requirement	Description
Base adaptations		
ExampleSensors::AssociatedSensor	Mandatory	See DSPxxxx.

Properties		
Antecedent	Mandatory	**Key**: Value shall reference the FanSensor (see X-7.4.7) instance 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**: *
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.

…

A.4.5 Examples of subclauses defining indication adaptations

Table A.7 details examples of subclauses within the "Adaptation" subclause of the "Implementation" clause that define specific adaptations of indications.

Table A.7 – Examples of subclauses defining specific indication adaptations

X-7.4.34 Adaptation: FanAddedAlert: CIM_AlertIndication

The FanAddedAlert indication reports the event that a fan was added to a computer system; for details, see the definition of message PLATMREG::PLAT0456.

The requirement level of the FanAddedAlert indication adaptation is conditional.

The implementation type of the FanAddedAlert adaptation is: "indication".

Condition: The FanLifecycleAlerts feature is implemented; see X-7.2.5.

Table X-45 lists the element requirements for this indication adaptation.

Table X-45 – FanAddedAlert: Element requirements

Element	Requirement	Description
Base adaptations		
Indications::AlertIndication	Mandatory	See DSP1054.
Alert messages		
PLATMREG::PLAT0456	Mandatory	See DSP8007.
Properties		
AlertingManagedElement	Mandatory	Value shall reference the Fan instance representing the added fan.
MessageID	Mandatory	Value shall match "PLAT0456".
OwningEntity	Mandatory	Value shall be "DMTF".
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-7.4.35 Adaptation: FanRemovedAlert: CIM_AlertIndication

The FanRemovedAlert indication reports the event that a fan was removed from a computer system; for details, see the definition of message PLATMREG::PLAT0457.

The requirement level of the FanRemovedAlert indication adaptation is conditional.

Condition: The FanLifecycleAlerts feature is implemented; see X-7.2.5.

The implementation type of the FanRemovedAlert adaptation is: "indication".

Table X-46 lists the element requirements for this indication adaptation.

Table X-46 – FanRemovedAlert: Element requirements

Element	Requirement	Description
Base adaptations		
Indications::AlertIndication	Mandatory	See DSP1054.
Alert messages		
PLATMREG::PLAT0457	Mandatory	See DSP8007.
Properties		
AlertingManagedElement	Mandatory	Value shall reference the Fan instance that represented the removed fan.
MessageID	Mandatory	Value shall match "PLAT0457".
OwningEntity	Mandatory	Value shall be "DMTF".
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.

X-7.4.36 Adaptation: FanFailedAlert: CIM_AlertIndication

The FanFailedAlert indication reports the event that a fan within a computer system failed; for details, see the definition of message PLATMREG::PLAT0458.

The requirement level of the FanFailedAlert indication adaptation is optional.

The implementation type of the FanFailedAlert adaptation is: "indication".

Table X-47 lists the element requirements for this indication adaptation.

Table X-47 – FanFailedAlert: Element requirements

Element	Requirement	Description
Base adaptations		
Indications::AlertIndication	Mandatory	See DSP1054.
Alert messages		
PLATMREG::PLAT0458	Mandatory	See DSP8007.
Properties		
AlertingManagedElement	Mandatory	Value shall reference the Fan instance representing the failed fan.
MessageID	Mandatory	Value shall match "PLAT0458".

OwningEntity	Mandatory	Value shall be "DMTF".
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.

X-7.4.37 Adaptation: FanReturnedToOKAlert: CIM_AlertIndication

The FanReturnedToOKAlert indication reports the event that a fan within a computer system returns to normal operation mode; for details, see the definition of message PLATMREG::PLAT0459.

The requirement level of the FanReturnedToOKAlert indication adaptation is optional.

The implementation type of the FanReturnedToOKAlert adaptation is: "indication".

Table X-48 lists the element requirements for this indication adaptation.

Table X-48 – FanReturnedToOKAlert: Element requirements

Element	Requirement	Description
Base adaptations		
Indications::AlertIndication	Mandatory	See DSP1054.
Alert messages		
PLATMREG::PLAT0459	Mandatory	See DSP8007.
Properties		
AlertingManagedElement	Mandatory	Value shall reference the Fan instance representing the fan that returned to normal operational state.
MessageID	Mandatory	Value shall match "PLAT0459".
OwningEntity	Mandatory	Value shall be "DMTF".
MessageArguments[0]	Mandatory	Value shall be identical to the value of the ElementName property in the CIM_Fan instance representing the fan that returned to the OK state.
MessageArguments[1]	Mandatory	Value shall be in WBEM URI format and refer to the CIM_ComputerSystem instance representing the scoping computer system.

X-7.4.38 Adaptation: FanDegradedAlert: CIM_AlertIndication

The FanDegradedAlert indication reports the event that a fan within a computer system starts operating in a degraded mode; for details, see the definition of message PLATMREG::PLAT0460.

The requirement level of the FanDegradedAlert indication adaptation is optional.

The implementation type of the FanDegradedAlert adaptation is: "indication".

Table X-49 lists the element requirements for this indication adaptation.

Table X-49 – FanDegradedAlert: Element requirements

Element	Requirement	Description
Base adaptations		
Indications::AlertIndication	Mandatory	See DSP1054.
Alert messages		
PLATMREG::PLAT0460	Mandatory	See DSP8007.
Properties		
AlertingManagedElement	Mandatory	Value shall reference the Fan instance representing the fan that is in a degraded state.
MessageID	Mandatory	Value shall be "PLAT0460".
OwningEntity	Mandatory	Value shall be "DMTF".
MessageArguments[0]	Mandatory	Value shall be identical to the value of the ElementName property in the CIM_Fan instance representing the failed fan operating in a degraded mode.
MessageArguments[1]	Mandatory	Value shall be in WBEM URI format and refer to the CIM_ComputerSystem instance representing the scoping computer system.

A.5 Example of the "Use cases" clause

Table A.8 provides an example of the "Use cases" profile specification clause.

Table A.8 – Example of "Use cases" clause

X-8 Use cases

…

X-8.3 DetermineFanState

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 2).

If the GetInstance( ) operation causes an exception, the use-case continues with step 4).

2) The client inspects the return value

- A return value of 0 indicates successful execution of the intrinsic operation; the use-case continues with step 3).
- A return value 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 value 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

The fan state could not be determined; reasons were reflected through either through the value of the return value or through CIM_Error instances delivered as part of an exception.

…

X-8.7 EnableFan

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 value:

   − A return value 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 value 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 value 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 value 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 value 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 value 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 value 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 value 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-8.7.4.3 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 lists 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-8.7.4.3 Failure with undefined state

The state of the fan is undetermined.

Annex B
(informative)

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.

The ABNF usage conventions defined in the Document conventions of this guide apply.

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 `SimpleChars`, 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 three-character sequence with the middle character not being `"a"` or `"u"`, 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, 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 [ CharacterExpression ]
```

A `CharacterExpression` is a descriptor for a sequence of one or more characters, for example:

`"X"`	matching `"X"` only,
`"ABC"`	matching `"ABC"` only,
`"ABC*"`	matching `"AB"`, `"ABC"`, `"ABCC"`, `"ABCCC"`, and so forth,
`"A[BC]D"`	matching `"ABD"` or `"ACD"`, or
`"1[.]{2,3}n"`	matching `"1..n"` or `"1...n"`.

Grouping

```
Grouping = "(" CharacterExpression ")" [ Multiplier ]
```

A `Grouping` is a `CharacterExpression` that optionally can be multiplied, for example:

`"(ABC)"`	matching `"ABC"`,
`"(XYZ)+"`	matching `"XYZ"`, `"XYZXYZ"`, `"XYZXYZXYZ"`, and so forth.

ChoiceElement

```
ChoiceElement = Grouping / CharacterExpression
```

Choice

```
Choice = ChoiceElement [ "|" Choice ]
```

A `Choice` is a choice from one or more `ChoiceElements`, for example:

`"(DEF)?"`	matching `""` or `"DEF"`,
`"GHI"`	matching `"GHI"`, or
`"(DEF)?|GHI"`	matching `""`, `"DEF"`, or `"GHI"`.

Left anchor

```
LeftAnchor = "^"
```

A `LeftAnchor` forces a match at the beginning of a string.

Right anchor

```
RightAnchor = "$"
```

A `RightAnchor` forces a match at the end of a string.

AnchoredExpression

```
AnchoredExpression = [ RightAnchor ] Choice [ LeftAnchor ]
```

An `AnchoredExpression` is a `Choice` that is optionally anchored to the left end, to the right end, or to both ends of a string.

AnchoredChoice

```
AnchoredChoice = AnchoredExpression [ AnchoredChoice ]
```

An `AnchoredChoice` is a choice from one or more `AnchoredExpressions`.

RegularExpressionInProfile

```
RegularExpressionInProfile = AnchoredChoice
```

A regular expression within a profile is an `AnchoredChoice`.

Annex C
(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 Deprecation of multiple inheritance for profiles Rules for the definition of indications Rules for defining the relationship to the managed environment Condensed structure of profile specifications Many clarifications and corrections
1.1.0m	2010-06-11	Work in progress release. Changes: Definition of metric-related requirements Definition of indication-related requirements DMTF adaptation diagrams
1.1.0n	2010-10-15	Work in progress release Changes: Many corrections and clarifications. Abstract profiles may reference DSP1033 Renamed the "Profile conventions for operations" subclause to "General requirements" Removed the following ABNF exceptions: Use of "|" in place of "/" for choices Use of ".." in place of "-" for ranges Insignificance of whitespace Removed events as profile element (covered with indications now) Revised version of the merge algorithm Combined all element requirements in one table, including base elements such as base adaptations Introduced state descriptions as profile element (primarily for use-cases) Introduced error reporting requirements as an extension of standard message requirements
1.1.0o	2010-12-17	DMTF Draft Standard

		Incorporated changes resulting from reviews: • Discourage use of "related profile" in favor of "referenced profile" • Divide referencing profiles into "profile derivation" and "profile usage" • Added requirement to specify operations using DSP0223 • Added definition of WBEM listener implementation conformance • Lowered the requirement for following the rules on when to use the "conditional" and "conditional exclusive" requirement levels, to a recommendation • Clarified allowable number of base profiles in a derived profile • Added requirement that the schema version of a derived profile is at least as recent as the most recent schema version of its base profiles • Clarified scoping relationship • Clarified which version of a profile is effectively referenced in a profile reference • Added provision to designate base adaptation candidates • Added rules for the repetition of schema requirements • Added provision for specifying requirements for instance creation and modification operations • Clarified that the PRP itself is exempted from the requirement that concrete profiles must reference the PRP • Lifted the requirement that state descriptions need to be named, for state descriptions defined within use cases • Lifted requirement to implement each used profile separately, and made that an implementation consideration • Adapted common text for "Terms and definitions" clause to the conventions set forth by the ISO/IEC Directives
1.1.0	2011-06-30	DMTF Standard Incorporated changes resulting from comments: • Refine the definition of requirement levels with respect to their impact on the implementation, and define how they are to be used in profiles • Synchronize the approaches for metrics and indications • Allow that indication/metric adaptations can also be defined on adaptations that are based on those in the Indications / Base Metrics profiles • Multiple alert message possible for one alert indication adaptation • Clarified that a business entity can be an "organization" • Introduce the concept of an implementation type for adaptations • Added the "prohibited" requirement level • Subcategories in the "Adaptation table" • Require that association adaptations, and adaptations they reference, are to be required separately in profiles, with the suggestion of defining a direct or feature based dependency • Allow concrete profiles to specify abstract adaptations (because those have no impact on clients or implementations) • Add provision to allow separate constraints to be specified for presentation, initialization and modification of properties • Add provisions to allow input value requirements for properties and method parameters • Prohibition of input values for key properties • Requiring profiles to define a CIM based discovery mechanism for conditional / conditional exclusive and optional profile elements that enables client to determine whether the profile element is implemented (see 7.5). • Lifted strong 20 word requirements in table cells to recommendation • Renamed "General requirements" subclause of "Adaptations" subclause to "Conventions" • Require a non-Null value for mandatory properties in adaptation instances (and for conditional / conditional exclusive properties, with the condition being True)

Bibliography

This clause lists references that are helpful for the application of this guide.

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

DMTF DSP1000, *Management Profile Specification Template 1.1*
http://www.dmtf.org/standards/published_documents/DSP1000_1.1.pdf

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