Document Number: DSP1001

Date: 2009-08-05

Version: 1.0.1

Management Profile Specification Usage Guide

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

CONTENTS

Figures

Tables

Foreword

DSP1001, *Management Profile Specification Usage Guide*, was prepared by the DMTF Profile Infrastructure Working Group.

Management Profile Specification Usage Guide

1 Scope

This guide specifies the usage and requirements of DMTF management profile specifications (see 5.1). Profiles may be specified in documents published by DMTF (see Annex B) or in specifications created by other organizations. In either case, a profile is specified in a set of specification divisions (in other words, sections).

The audience for this guide is anyone creating a specification including DMTF profiles.

A profile specification generally follows this form:

a) Organization-specific front matter

b) Organization-specific, non-profile clauses

c) One or more profiles (defined in this guide)

d) Organization-specific, non-profile clauses

e) Organization-specific annexes

The majority of this guide addresses “c. One or more profiles”.

1.1 Profiles published by DMTF

Annex B specifies the format that shall be used for management profile specifications. The standard DMTF specification format applies to profile specifications.

1.2 Profiles published by other organizations

Other organizations should create their own guidelines for profiles, but the requirements of this guide (other than Annex B) shall be adhered to. This specification defines a set of specification divisions for profiles. An organization may opt to “demote” these divisions to a different heading level. For example, “6. Synopsis” may become “8.6 Synopsis” or “8.2.6 Synopsis”. However, the relative heading numbering shall be maintained (for example, all headings shall be demoted identically), and all template divisions shall be provided. This allows another organization to embed profile specifications in a larger document while preserving a recognizable profile format for readers.

This guide is **not** a template for a profile specification. To create a profile specification, start with the publishing organization’s template and add divisions as described in this guide.

This guide is **not** a profile specification; it defines the requirements for creating a profile specification.

Certain words and terms used in this guide have a specific meaning beyond the normal English meaning. These words and terms are defined in clause 3 ("Terms and Definitions").

2 Normative References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

DMTF DSP0004, *CIM Infrastructure Specification 2.5,*
http://www.dmtf.org/standards/published_documents/DSP0004_2.5.pdf

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

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

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

IETF RFC3629, *UTF-8, a transformation format of ISO 1064*6, Nov 2003
http://tools.ietf.org/html/rfc3629

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

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

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

3 Terms and Definitions

For the purposes of this document, the following terms and definitions apply.

3.1
CIM element
CIM classes (including associations and indications), properties (including references), or methods

NOTE: For the purpose of this document, CIM qualifiers and schemas are not considered CIM elements.

3.2
clause
the basic (that is, “top-level”) component in the subclause of the content of a document

NOTE: The clauses in each document or part shall be numbered with Arabic numerals, beginning with 1 for the “Scope” clause. (See ISO/IEC Directives, Part 2.)

3.3
client
CIM client

3.4
conditional
keyword that describes items that are required under specified conditions

NOTE: See 5.3.1 for requirements of specification of conditional elements.

3.5
deprecated
keyword indicating that an element or profile behavior has been outdated by newer constructs.

NOTE: Deprecated elements may become obsolete in future versions of the profile. Authors should avoid using deprecated elements and attributes. Server implementations should continue to support for reasons of backward compatibility.

3.6
division
a component (that is, section) of a specification

3.7
mandatory
keyword that describes items that are required under all conditions

3.8
match
(CIM property values) keyword indicating that a property is equal to one or more values

3.9
may
keyword that indicates flexibility of choice with no implied preference

3.10
may not
keywords that indicate flexibility of choice with no implied preference

3.11
optional
keyword that describes items that are not required

NOTE: See 5.3.1 for requirements of specification of optional elements

3.12
organization
consortium, standards group, or company creating a DMTF profile specification

3.13
paragraph
unnumbered subdivision of a clause or subclause (See ISO/IEC Directives, Part 2.)

3.14
pattern
(CIM property values) The value of the property shall follow the supplied pattern

3.15
server-side implementation
CIM providers

3.16
shall
keyword indicating a mandatory requirement

3.17

should

keyword indicating flexibility of choice with a preferred alternative; equivalent to the phrase "it is recommended"

3.18

subclause

a numbered subdivision of a clause

NOTE A primary subclause (for example 5.1, 5.2, etc.) may be subdivided into secondary subclauses (for example 5.1.1, 5.1.2, etc.), and this process of subdivision may be continued as far as the fifth level (for example 5.1.1.1.1.1, 5.1.1.1.1.2, etc.). See ISO/IEC Directives, Part 2.

4 Symbols and abbreviated terms

For the purposes of this document, the following abbreviated terms and definitions apply:

4.1

UfcT

User Friendly class Tag (see DSP0215)

4.2

UfiT

User Friendly instance Tag (see DSP0215)

5 Specifying management profiles

5.1 Profile terminology

A profile is a specification that defines the CIM model and associated behavior for a management domain. The CIM model includes the CIM classes, associations, indications, methods and properties. The management domain is a set of related management tasks. A profile is uniquely identified by the name, organization name and version.

It is desirable to break up complex management domains into sets of profiles. This allows reuse of profiles in different contexts and also allows decomposition of a complex management domain to help readers.

- **An autonomous profile** defines an autonomous and self-contained management domain. This includes profiles that are standalone, or have relationships to other profiles.
- **A component profile** describes a subset of a management domain. A component profile includes CIM elements that are scoped within an autonomous profile (or in rare cases, another component profile). Multiple autonomous profiles may reference the same component profile.

A complete management domain may often be expressed as a combination of an autonomous profile with a collection of component profiles. Typically, an autonomous profile includes a computer system and component profiles do not. The elements of the component profiles are typically associated to a System instance in an autonomous profile – referred to as the **scoping profile**. Optimally, profiles are defined in such a way that allows a component profile to be scoped to different autonomous profiles.

The following are examples of combinations of profiles. (At the time of this writing, there are no compliant profile specifications to reference; these examples are hypothetical and will be replaced by actual examples as they are developed).

- **Autonomous profile with optional component profiles** – Embedded control systems optionally include management interfaces for fans or power supplies. Elements related to core control system interfaces are included in the autonomous profile. The fan and power supply elements are in separate component profiles.
- **Multiple autonomous profiles sharing component profiles** – Disk arrays and volume managers provide similar RAID virtualization capabilities from a device of host-resident software. The RAID virtualization component profile is shared by, but mandatory for the Array (external virtualization hardware) and Volume Manager (host-resident virtualization software) autonomous profiles.
- **Related component profiles, scoped to the same autonomous profile** – Many types of systems include batteries — sometimes batteries are configured in redundant sets. This could be modeled as a battery component profile with a separate, optional battery redundancy component profile. Elements of component profiles are scoped to a System instance defined in the context of a top-level autonomous profile in the scoping hierarchy.
- **Scoping between component profiles** – In some cases, CIM defines scoping between non-system elements. For example, ServiceStatisticalInformation is scoped to a Service, which is then scoped to a system.

A **specialized profile** is based on and constrains another profile specification. An **abstract profile** specifies common elements and behavior that form the base for specialized profiles. For example, a storage statistics abstract profile could be specialized for either direct disk storage or shared filesystems. A specialized profile may be either an autonomous or a component profile; an abstract profile may be either an autonomous or a component profile.

Abstract profiles shall not be implemented or deployed; they serve as templates for specialization. Abstract profiles may contain model and behavior. See Annex D for additional information about profile specialization.

5.2 Profile Registration

The CIM schema includes a model for profiles, including the RegisteredProfile class, and ElementConformsToProfile and ReferencedProfile associations. See DSP1033 for details.

Profiles other than the *Profile Registration Profile* shall include the *Profile Registration Profile* as a mandatory component profile.

5.3 Requirements for non-mandatory elements

Profiles may include non-mandatory CIM elements. These elements shall reflect optional or conditional behavior in the management domain. For example:

- Some host controllers have status LEDs that can be blinked from the management software — making it easy to locate the card. This is not considered a core capability of host controllers, but implementations that provide this capability should use the model described in the profile.
- Some management domains have either-or behavior. The same interfaces for storage volume creation are used in device-based RAID arrays (resulting in StorageVolume instance) and host-based volume managers (resulting in a LogicalDisk instance). A profile may be defined to support either StorageVolume or LogicalDisk. A scoping profile may constrain the instances of a component profile to be of one type or the other.
- The standard model may not be appropriate for all implementations. CIM models file systems with FileSystem associated to StorageExtent using ResidesOnExtent associations. For many filesystems, the relationship to underlying disk storage may be decomposed using BasedOn associations. The same model could be used with journaled filesystems, but the BasedOn

associations may change too rapidly (many times a second) to be useful — so a filesystem profile may want to treat BasedOn as optional.

5.3.1 Best practices for non-mandatory elements

This guide distinguishes between three levels of requirements:

Mandatory – CIM elements marked as mandatory shall be supported by the server-side implementation. Clients can rely on their support, once they have determined that the profile is supported. An implementation confirming to a profile specification shall support a mandatory item as defined by the specification.

Optional – CIM elements marked as optional may be supported by the server-side implementation. If supported, optional elements shall be supported as specified. The specification shall describe a CIM-based technique that conformant implementations shall support to enable a client to determine when an optional item is present.

Conditional – CIM elements marked as conditional may be supported by the server-side implementation. Conditional elements are mandatory if some CIM-based and domain-relevant condition is met. If supported, conditional elements shall be supported as specified. The specification shall describe a CIM-based technique that conformant implementations shall support to enable a client to determine when a conditional item is present.

A profile specification may include descriptions of generic techniques for CIM-based discovery of optional elements. For example, “A client may determine support for optional properties by getting an instance; if the property is returned, it shall be supported as defined in this specification.”

Profile specifications with conditional dependencies on other profiles should avoid duplicating specification. If a property in a class defined in one profile is constrained in a second profile, then non-constrained properties shall not be duplicated in the second profile. For example, the Battery profile specification has a conditional dependency on the Sensor profile. The contents of the Sensor profile should not be duplicated in the Battery profile unless the Battery needs to add further constraints, and then only the element (property or method) with the additional constraints should be included in the Battery profile.

If there is a scoping relationship, the component profile shall specify the spanning association. If there is not a scoping relationship, then the spanning association shall be owned by one profile and may be constrained by the peer profile.

5.3.2 Specification of conditional behavior

Each area of conditional behavior should be documented as a separate subclause of the Implementation division. There are two aspects to specifying conditional behavior: the condition and the conditional behavior.

One of the following techniques shall be used to specify a condition:

- **Another profile registered through CIM_RegisteredProfile**. For example, the Fan Profile requires a specific element only if the Sensor Profile is also registered. There are two ways the existence of one profile influences another profile: associations that flow between two profiles and constraints in one profile of elements defined in a separate profile. If the RegisteredProfile is instantiated and associated (perhaps via some scoping element), the referenced element(s) shall be supported.
- **A capabilities property** (generally in a CIM_Capabilities class instance) that tells the client whether conditional behavior is supported. This property and the class containing this property are mandatory. If the property is set as specified, the referenced element(s) shall be supported.

- **Existence of a class that has a static lifespan** (in other words, is instantiated for the entire life of the supporting server-side implementation). For example, a conditional behavior is tied to the existence of a CIM_Service instance associated with a scoping system and the static class (CIM_Service in this example) is mandatory.
- **Association to a class with identical lifespan** – These are cases where one class is guaranteed to exist whenever a second class exists – or not at all. For example, if supported, AlarmDevice is associated via AssociatedAlarm to a PortController. A profile may state that if the PortController supports an interface for a client to blink an LED, AlarmDevice and AssociatedAlarm shall be instantiated. A client discovers support for this conditional behavior by looking for the AssociatedAlarm association.

The conditional behavior is the set of CIM elements that are considered mandatory if the condition is met. The profile specification documents the condition and the condition elements and semantics that are related to the condition.

5.4 Associations and superclasses

In some contexts, associations reference classes that are superclasses of those used in a particular profile. When used in Object diagrams of a profile, these associations may be linked to instances that are subclasses of the association ends. For example, the MOF definition of SystemDevice includes references to System and LogicalDevice; but a profile may specify that instances of SystemDevice associate instances of System to instances of DiskDrive. Profile specifications shall not include superclasses solely for the purpose of satisfying association references. Profile specifications may include superclasses where they model the management domain. For example, a profile may include both StorageExtent (a concrete superclass) and LogicalDisk (one of its subclasses).

5.5 Requirements for normative descriptions of CIM elements

Normative text for CIM classes, indications, and properties may appear in the “Description” column of the class tables in the “CIM elements” division or in the "Implementation" division. The text in the "Description" cell should be short: no longer than 20 words. Text longer than 20 words should be placed in a subclause of the "Implementation" division and a cross reference (for example, “see 4 5 6 7”) placed in the "Description" cell.

5.6 Specification of constraints to elements from related profiles

A profile may constrain elements of related profiles. For example:

- Properties of related profiles may be constrained (for example, when used as a component profile of the current profile, a subset of values are valid);
- Classes of related profiles may be constrained by specifying a subclass;
- Methods of related profiles may be constrained by this profile (for example, specify a subclass of a parameter in an extrinsic method from a related profile).

Other constraints to elements in related profiles are possible.

These constraints shall be specified in the "CIM elements" division (see 6.6) or "Implementation" (see 6.3) division, contingent on “the rule of 20 words” described in 5.5.

5.7 Backwards compatibility, deprecated elements and behavior

A profile specification shall be compliant to previous minor versions of the profile. In other words, version 2.4 of a profile shall be compliant to versions 2.0, 2.1, 2.2, and 2.3 of the profile. A new minor version may extend the functionality of previous versions, but it shall maintain all the requirements of previous versions. Incompatible changes require incrementing the major version number.

Compliance to a profile specification requires compliance to previous minor versions of the profile with the same major version number. In other words, compliance to version 2.4 of a profile also requires compliance to versions 2.0, 2.1, 2.2, and 2.3 of the profile. Each updated profile specification shall document deprecated properties that were part of previous minor versions of the profile with the same major version.

Each profile specification shall be self-contained and not require the reader to reference previous versions of the profile specification. All deprecated properties and classes shall be documented in the "CIM elements" division. However, a profile specification may refer to a diagram from a previous version to visually assist the reader in understanding the model including deprecated elements.

In order to support backwards compatibility, a profile specification may include standard and deprecated variants of the same element. In this case, the deprecated elements shall include the word "Deprecated" in the "Description" column.

5.8 Diagrams

Three types of diagrams are commonly used in profile specifications:

- **Class diagrams** provide a view of the classes of a profile (and possibly related classes in other profiles)
- **Object diagrams** (also referred to as Instance Diagrams) provide a view of a set of related class instances at a point in time. Object diagrams may be associated with use cases, showing how the use case affects properties and classes.
- **Sequence diagrams** show the interaction between classes in terms of methods and operations.

5.8.1 General diagram guidelines

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

A diagram shall not mix the conventions of class and object diagrams.

The diagrams shall follow DMTF diagram conventions described in Annex E.

The “CIM_” prefix shall be left off a class name if it refers to a class in MOFs defined by DMTF. Prefixes should be added if the profile is specifying classes that extend the MOFs defined by DMTF.

Fonts in diagrams should be 12 point and shall not be less than 10 point.

5.8.2 Class diagrams guidelines

Class diagrams convey profiled classes and associations.

Methods or properties shall not be included in profile specification class diagrams (these types of diagrams are informally called model diagrams). Eliminating properties and methods in these diagrams eliminates the risk that these elements are specified differently in the diagram and the text format included in profile specifications. In other contexts, class diagrams may include these elements; the guidelines here apply to class diagrams in profile specifications.

Classes are represented with the two-horizontal-compartment box. The top compartment contains the class name.

Classes that are owned by the profile leave the lower compartment empty.

Abstract classes are generally not represented in class diagrams. There are a few exceptions:

- profiles that are defined as “abstract” (as described in 6.1) may reference abstract classes and include them in diagrams
- when the object in the current profile is represented by a subclass or superclass in a related profile
- when an abstract class is used to represent one of several possible concrete subclasses

Inheritance shall only be represented if the profile specifies use of a class and its superclass. See 5.4.

When a profile makes use of a class that is defined/owned by another profile, text in parenthesis in the lower compartment identifies the profile where the class is defined.

Cardinality shall be indicated for all associations. In the case where a profile further restricts cardinality on the associations as documented in the MOF (that is, instead of 0-n, the profile requires 1-n), the cardinality defined in the class diagram shall reflect the additional restrictions specific to the profile.

Associations shall not include properties or methods. If the association is defined in another profile, a parenthetical reference to the other profile is required.

Each class diagram shall have a label formatted as follows: "<profile name>: Profile Class Diagram".

5.8.3 Object diagram guidelines

Object diagrams demonstrate an example instantiation and ideally are illustrative of best practice implementations.

The names of objects should be specified using the following format:

```
Instance Name : Class Name
```

If the instance is not ambiguous, the object name may be abbreviated as a class name without the instance name.

```
: Class Name
```

Abstract classes and inheritance are not represented in object diagrams. If a variety of concrete classes may be substituted for an abstract class, make an object diagram using one concrete class and provide explanatory text with the diagram pointing out the other concrete classes that are applicable.

Instances are represented with a two-horizontal-compartment box. The top compartment contains the class name. The bottom compartment contains applicable properties that are needed to be illustrative.

All applicable properties that are needed to be illustrative of the instance data requirements are to be listed, followed by a space, colon, space and an example value for the property.

Methods should not be included in object diagrams.

Note that properties and methods shall be listed and documented in the class tables (see 6.6.2). Including them in the diagrams is duplication; the profile author shall keep these up-to-date with changes in the rest of the profile and the MOFs.

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

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

Associations that have properties and/or methods that are illustrative of the instance data are to be listed below the association class name.

5.8.4 Sequence diagrams

Sequence diagrams depict the interaction between class instances, in the form of method calls and call returns.

The names of objects should be specified using the following format:

```
Instance Name : Class Name
```

If the instance is not ambiguous, this may be abbreviated as a class name without the instance name.

```
: Class Name
```

5.8.5 Deprecated CIM elements in diagrams

Diagrams in profiles should not include deprecated CIM properties. Diagrams may include deprecated classes; they should follow current DMTF diagram guidelines (see Annex E).

6 Profile specification divisions

Each profile specification shall include the divisions shown in Table 1.

Table 1 – Divisions of a profile specification

Synopsis	See 6.1.
Description	See 6.2.
Implementation	See 6.3.
Methods	See 6.4.
Use cases	See 6.5.
CIM elements	See 6.6.

The usage of these profile divisions is explained in detail below.

6.1 Profile division “Synopsis”

This division starts with the profile’s name, organization, and version number, formatted as follows:

Profile name: <profile name>

Version: <version>

Organization: <organization name>

CIM Schema Version: <CIM schema version>

Central Class: <CIM Class Name>

Scoping Class: <CIM Class Name>

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

The version number shall follow the same rules as the CIM schema version numbers (see DSP0004, 2.3.1 Schema Versions).

The organization name shall be the name of the organization that is publishing the profile. For profile specifications published by DMTF, the organization name shall be “DMTF”.

The CIM schema version shall be the earliest CIM schema version that meets the requirements of the profile.

The class names for scoping and central classes are required. A central class is used in a profile as the focal point for identifying conformance with that profile. Instances of this class will be associated via ElementConformsToProfile associations to the instance of RegisteredProfile advertising conformance for the profile. A scoping class is used in a profile as the focal point for identifying conformance with that profile.

If the current profile is a component profile, it shall include a line (after "Scoping Class") formatted as follows:

Scoping Algorithm: <scoping algorithm>

The scoping algorithm shall define the association traversal path from a central instance to its scoping instance. The association traversal path shall be specified in prose as a list of associations and associated classes, along with restrictions where required. Typically specifying a single association class is sufficient.

For more information about central classes and scoping classes, see DSP1033. In addition to the class names, this division may include text describing the usage of central and scoping classes.

If the current profile specializes another profile, it shall include a line (after the Schema version) formatted as follows:

Specializes: <organization-name> <profile-name> <version-name>

This may include a list of organization/profile/version names separated by commas. See Annex D for more information about profile specialization.

If the current profile is defined to be “abstract” and specializations must be used in implementations, it shall include the following statement:

“This abstract profile specification shall not be directly implemented; implementations shall be based on a profile specification that specializes the requirements of this profile.”

This division shall include a one-paragraph summary that may be used in other documents to describe the profile. This paragraph shall state whether the profile is an autonomous or component profile (see 5.1).

If the current profile is not abstract, the summary shall be followed by a table of related profiles. The name, organization and version of each related profile shall match those from the "Synopsis" section of the related profile’s specification.

The “Requirement” cell shall specify whether the related profile is mandatory, optional, or conditional. If the related profile is optional, the "Requirement" cell shall consist of the word “Optional”. For optional or mandatory profiles, the “Description” cell should provide a short description of the related profile and its relationship to the current profile, or a cross reference to a subclause containing this information.

If the related profile is conditional, the "Requirement" cell shall consist of the word “Conditional”. The condition (including an algorithm for determining whether the condition is met) shall be specified in text and may be specified in a programming language. Subject to the requirements in 5.5, the “Description” cell contains this specification of the condition, or a cross reference to a subclause containing this specification of the condition.

If the profile is specialized, it shall include a row for the parent profile and the "Requirement" cell shall contain “Specializes”. A profile that is parent to a specialized profile and not implemented directly shall not be registered (via ElementConformsToProfile to RegisteredProfile).

If the profile has no related profiles, this division shall contain “Not defined in this standard” rather than the related profiles table. This applies to abstract profiles and the *Profile Registration Profile*.

Table 2 provides an example of a "Synopsis" division.

Table 2 – "Synopsis" division example

1. Synopsis

Profile name: Power Supply

Version: 1.3.0

Organization: DMTF

CIM schema version: 2.9

Central class: CIM_PowerSupply

Scoping class: CIM_ComputerSystem

Scoping algorithm: CIM_SystemDevice

The *Power Supply Profile* is a component profile that extends the management capability of referencing profiles by adding the capability to describe power supplies.

Table 3 – Related profiles table example

Profile Name	Organization	Version	Requirement	Description
Battery	DMTF	1.1.0	Optional	
Battery diagnostics	DMTF	1.0.0	Conditional	See 6.2.1
Sensor	DMTF	1.0.0	Mandatory	See 6.2.2

6.2 Profile division “Description”

The "Description" division describes the management domain implemented by this profile and provides an overview of the model. The "Description" division shall not include normative documentation. This division should describe how the classes of the profile relate to the management domain.

This division should contain class diagrams (see 5.8.2).

6.3 Profile division “Implementation”

The "Implementation" division shall contain requirements of the model that are not covered by other divisions (such as the "Methods" division) and guidelines related to implementation. The profile author may include requirements here and reference them from other divisions, and describe the relationship between the model and underlying instrumentation.

Profile authors may choose to partition the information in this division into sub-topics. This division may contain informative text to introduce these sub-topics. The sub-topics may be based on domain behaviors that apply to multiple CIM elements (for example, “Element discovery”) or may be based on specific CIM elements. A domain behavior subclause may describe (and be referenced by) several CIM elements.

Table 3 is an example of an "Implementation" division.

Table 3 – "Implementation" division example

2.3 Implementation
The Host Controller profile consists of three areas of functionality – basic element inventory, baseboard management, and redundant controllers

2 3.1 Basic element inventory implementation
Some informative text about basic element inventory.

Basic element inventory requires AdminDomain, TCPProtocolEndpoint, and HostedAccessPoint. TCPProtocolEndpoint.PortNumber shall be 12345.

2.3.2 Baseboard management implementation
Baseboard management requires Sensor and mandatory support for the *Fan Profile*.

The "CIM elements" division also contains normative information, but that information is presented in tables. If the requirements do not readily fit in a table format, they shall be specified in this division and cross-referenced from the tables (see 5.5).

Note that cross-references shall target numbered divisions. Any requirements referenced elsewhere shall appear as separate subclauses of the "Implementation" division.

Restrictions on the multiplicity of instances in a profile or associated to some other class may be specified in this clause.

Conditional behavior shall be specified as subclauses of the "Implementation" division. Each conditional behavior subclause includes one or more paragraphs that provide text describing the conditional behavior and may include diagrams. Each conditional behavior subclause shall specify the condition (the CIM-based mechanism a client uses to determine if the conditional behavior is supported by the implementation) and the conditional behavior (the CIM elements and semantics that are mandatory if the condition is met).

The condition is defined in terms of one or more of the techniques for specifying conditional behavior (see 5.3.1). Suggested wording includes:

- If ProtocolControllerMaskingCapabilities.ProtocolControllerSupportsCollection is TRUE, the implementation shall support SystemSpecificCollection and MemberOfCollection referencing StorageHardwareID instances as depicted in figure 200. If ProtocolControllerMaskingCapabilities.ProtocolControllerSupportsCollection is FALSE, the implementation shall not have any MemberOfCollection instances referencing SystemSpecificCollection and StorageHardwareID instances.
- If persistent binding is supported, the implementation shall include a single instance of StorageNameBindingService associated to System via a HostedService association.

The elements that depend on this condition may be classes, associations, properties, methods, or indications. If these CIM elements are only specified when the condition is met, they shall be documented in the conditional behavior subclause. If CIM elements are specified whether on not the condition is met, they shall be documented elsewhere in the "Implementation" division and also documented in the conditional behavior subclause if their behavior changes when the condition is met.

Table 4 is an example of specification of conditional behavior in "Implementation" and "CIM Element" divisions.

Table 4 – Example with conditional behavior

10. Implementation

10.1 Battery support
Battery support is optional. If the underlying controller includes batteries and provides the ability to determine basic status and asset information, the vendor should also implement the Battery Profile. If the Battery Profile is implemented, the implementation shall also support the SytemDevice associations from the controller's System to each CIM_Battery instance.

If the underlying implementation provides the capability to test the health and expected lifespan of batteries, the Battery Diagnostics Profile should also be supported. If so, then the implementation shall also support HostedService associations between the controller's System and DiagnosticService.

10.2 LED blink
Implementations may optionally support LED blinking by instantiating an AlarmDevice instance and associating it via AssociatedAlarm to Port instances. The server-side implementation shall support the SetAlarmState method on AlarmDevice. If the LED uniquely identifies a simple port on a multi-port controller, then AssociatedAlarm should reference a PortController. Otherwise, it shall reference an FCPort.

….

13. CIM elements
…

13.2 CIM_AlarmDevice
Conditional: see 10.2

13.2 CIM_AssociatedAlarm
Conditional: see 10.2

Table 32 – CIM_AssociatedAlarm

Element	Requirement	Description
Antecedent	Mandatory	Reference to the AlarmDevice
Dependent	Mandatory	Reference to the FCPort or PortController. See 10.2

6.4 Profile division "Methods"

The "Methods" division provides a list of methods supported by this profile — in other words, methods of the classes of the profile. Profile usage of both extrinsic methods and generic operations (for example intrinsic methods) are included, but the specification formats are different. Table 5 depicts the general look of a "Methods" division of a profile specification. The class names shall include the prefix (for example, "CIM_ ").

This division may include sequence diagrams or state diagrams that relate to the methods.

Table 5 – Overview of "Methods" division

4 Methods
4.1 CIM_ClassName ExtrinsicMethod1
(see Management Profile Usage Guide 6.4.1)
4.2 CIM_ClassName ExtrinsicMethod2
(see Management Profile Usage Guide 6.4.1)
… additional extrinsic methods …
4.3 Profile Conventions for Operations
(specifies the approach used in the particular profile specification for documenting operations; see Management Profile Usage Guide 6.4.2)
4.4 CIM_ClassName1 Operations
(see Management Profile Usage Guide 6.4.3)
4.5 CIM_ClassName2 Operations
(see Management Profile Usage Guide 6.4.3)
… additional per-class operations …

Note that management profile specifications describe use of operations for manipulating class and association instances and shall not specify operations that manipulate schemas and qualifiers. Constraints to methods in related profiles shall not be placed in this division; see 5.6.

6.4.1 Extrinsic Methods

Each extrinsic method of the profile shall be specified in a separate subclause of the "Methods" division. The subclause may include a short (one or two paragraph) description and tables specifying the return values and parameters. Text from MOF descriptions may be used to describe extrinsic methods, but shall be reworded as standard English sentences. It is not required that descriptions be provided in the profile if the MOF descriptions are clear and appropriate for the profile. In this case, the profile shall reference the MOF. This subclause may include references to use cases (see 6.5) for tasks that include this method.

If the profile specifies use of standard messages, this subclause shall include a table specifying standard messages. If the profile does not specify use of standard messages, no table shall be included, but the description should state “No standard messages are defined.” The standard messages table has two columns. The left column contains a return value in parenthesis followed by the name of the registering organization and the message ID from that organization. The right column contains the message text (abbreviated, if appropriate).

If the MOF descriptions for return values adequately describe the return values as used in this profile, then this subclause should reference the MOF (for example, “See the return values for ModifySyncronization in the MOF for CIM_StorageService.”). If the MOF descriptions for return values need to be clarified for use in this profile, then they shall be specified in a table. The return values table has two columns: "Value" and "Description". The return values shall be formatted at the numeric valuemap followed by the string value in parenthesis. For example: “1 (Not Supported)”.

If the MOF descriptions for method parameters adequately describe the methods as used in this profile, then this subclause shall reference the MOF. If the descriptions of methods or method parameters from the MOF need to be clarified, they shall be specified in a table.

The parameters table has four columns: "Parameter qualifiers", "Name", "Type", and "Description". Unlike MOF usage, IN is included in this table if IN is true and shall not be included if IN is false. OUT is included in this table if OUT is true and not included if OUT is false. The "Qualifiers" column also includes “REQ” if the parameter includes the REQUIRED parameter in the MOF. A profile specification shall not change interpretation of MOF qualifiers, just present IN, OUT, and REQ to help the reader understand the use of these parameters.

If the return values or messages relate to conditions, this shall be specified in the Description (for example, “only valid if the optional sensor profile is supported”).

The text in the "Description" or "Description/Value" cells should be short; no longer than twenty words. Text longer than twenty words should be placed in a subclause of “Extrinsic Methods” and referenced from the table cell.

Table 6 includes an example of a subclause for an extrinsic method.

Table 6 – "Extrinsic Methods" division example

4.1 CIM_StorageService ModifySynchronization Extrinsic Method

This method modifies (or starts a job to modify) the synchronization association between two storage objects. If 0 is returned, the function completed successfully and no ConcreteJob instance was created. If 0x1000 is returned, a ConcreteJob was started and a reference to this Job is returned in the Job output parameter. A return value of 1 indicates the method is not supported. All other values indicate some type of error condition.

ModifySynchronize errors are specified in Table 36, and parameters are specified in Table 37.

Table 36 – ModifySynchronization method standard messages

(return) MessageID	Message
(5) SNIA.DRM24	Invalid Transition State
(4) SNIA.MP2	Operation not supported

Table 37 – ModifySyncronization Method Parameters

Qualifiers	Name	Type	Description/Values
IN, REQ	Operation	uint16	Type of operation to modify the replica: 2 (Detach) 3 (Fracture) 4 (Resync)5 (Restore) 6 (Prepare)
OUT	Job	CIM_ConcreteJob REF	Returned if job started.
IN, REQ	Synchronization	CIM_StorageSynchronized REF	Association to replica that will be modified

Extrinsic method names are also listed in property tables; see 6.6.2.

6.4.2 Generic Operations

The generic operations divisions consist of a subclause describing conventions followed by subclauses that describe each operation with profile-specific behavior.

Including separate subclauses for each operation for each class may produce many divisions containing the same boilerplate text. Many profiles require most operations to follow the behavior in the operations specification; other profiles may expect many exceptions. In order to allow profile authors to optimize profile text, they are allowed to make a choice between three different options as described in Table 7.

6.4.2.1 "Profile conventions for operations" subclause

The profile author’s conventions for specifications and requirements for operations shall be specified in a separate subclause of the "Methods" division. This subclause follows subclauses for extrinsic methods (if any).

This subclause shall contain text describing which operations specifications were considered when the profile specification was developed. For example, "This profile specification defines operations in terms of DSP0200 (CIM Operations over HTTP)." If a replacement generic operations specification is available, it should be referenced in this subclause instead of DSP0200.

This subclause shall contain text defining the conventions used to specify each class’s requirements for operations. Three options are possible; this subclause shall include one of the paragraphs from Table 7.

Table 7 – Profile convention options

Option	Text to place in "Profile conventions for operations" division
Option 1 – table includes each operation for each class	Deprecated; now covered by option 2, with additional requirements specified in 6.4.2.2. “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 class specific subclauses of the "Methods" clause.	The "Profile conventions for operations" subclause of the Methods clause shall contain the following 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 this clause." A profile may define a default list of operations, as follows: "The default list of operations is as follows: operation-1 operation-2 …" The default list may be extended for classes referenced by an association, as follows: "For classes that are referenced by an association, the default list of operations includes the following operations in addition: a-operation-1 a-operation-2 …" The applicability of the default list shall be specified in class specific subclauses of the "Methods" clause; see 6.4.2.2.
Option 3 – table includes operations with profile-specific requirements. Other operations may be implemented.	Deprecated; now covered by option 2, with additional requirements specified in 6.4.2.2. “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 operations is typically the operations related to manipulation of instances and possibly operations to execute queries.

There is no requirement to specify usage of the InvokeMethod operation; it is implicitly required if the profile defines any extrinsic methods.

6.4.2.2 Per-class operations subclauses

A subclause shall be included for each class (including associations) of the profile.

If a default list of operations is defined in the "Profile conventions for operations" subclause (see 6.4.2.1), and the default list shall apply unmodified for the specified class, the following statement shall be provided:

> "All operations in the default list in <pco-num> shall be implemented as defined in <op-spec>.
>
> NOTE: Related profiles may define additional requirements on operations for the profile class."

If a default list of intrinsic operations is defined, or if additional operations are specified for the specified class, a table shall be provided that details implementation requirements for specific operations that are not covered by the default requirement. The table shall be preceded by the statement:

> "Table <table-num> lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in <op-spec>. In addition, and unless otherwise stated in Table <table-num>, all operations in the default list in <pco-num> shall be implemented as defined in <op-spec>.
>
> NOTE: Related profiles may define additional requirements on operations for the profile class."

A profile may use the requirement level "Unspecified" to state that no requirements are defined by the specified profile for a particular operation listed in the default list.

If a default list of intrinsic operations is not defined, a table shall be provided that lists each intrinsic operation that is required to be implemented by the specified profile. The table shall be preceded by the statement:

> "Table <table-num> lists implementation requirements for operations. If implemented, these shall be implemented as defined in <op-spec>.
>
> NOTE: Related profiles may define additional requirements on operations for the profile class."

The variables in all these statements shall be resolved as follows:

<table-num>:	The number of the table
<op-spec>:	A reference to the operations specification
<pco-num>:	The subclause number of the "Profile conventions for operations" subclause.

If a table is provided, it shall include columns for "Operation Names", "Requirements", and "Messages". The interpretation of the table depends on the statement preceding it as defined in this subclause.

Table 8 demonstrates the format for a table of operations (this format applies to any of the three options).

Table 8 – Per-class operations requirements example

Operation	Requirements	Messages
CreateInstance		
DeleteInstance		
EnumerateInstance		
EnumerateInstanceNames		
GetInstance		

The valid options for the "Requirements" column depend on the selected option described in 6.4.2 and may include the following values:

Mandatory – Compliant implementations shall support this operation either per the referenced operations specification or per the requirements of this specification. The implementation shall not return CIM_ERR_METHOD_NOT_FOUND or CIM_ERR_METHOD_NOT_AVAILABLE when the method is invoked for the target class.

Optional – Compliant implementations may support this operation either per the referenced operations specification or per the requirements of this specification.

Unspecified – Support for this operation is not specified by this specification. A client should not assume that the operation is implemented or if implemented, that it is implemented as defined in the referenced operations specification.

If the profile defines behavior for operations different than what is described in the referenced operations specification, the operation shall be documented in a separate, numbered subclause of the Methods division with a heading containing the class and operation names. The heading is followed by text that describes the requirements for the operation – including all side effects to the model and the expected results in the underlying instrumentation. This subclause shall be referenced from the "Requirements" column for the appropriate operation.

The "Messages" column may include a reference to a table of standard messages (using the format in Table 6). This table shall be left blank if the message usage of the profile matches the usage described in the referenced operations specification.

When a profile identifies ModifyInstance as supported for a class, the default interpretation is that all non-key properties on the class may be modified using the ModifyInstance implementation. If a profile wishes to require that certain properties are modifiable or are not modifiable, explicit normative text to that effect needs to be included as a subclause of the operations clause for the class.

There is no requirement to specify usage of the InvokeMethod operation; it is implicitly required if the profile defines any extrinsic methods.

6.4.2.3 Operations related to associations

Operations that enumerate instances (or instance names) relative to a particular object shall be specified in the per-class operations division for the class representing the originating endpoint of the operation, not the association class. For example, when specifying operations as defined in DSP0200 in SystemDevice's operation table, Associators and AssociatorNames would be specified as "shall not be supported," but System and LogicalDevice operation tables would include Associators and AssociatorNames. Similarly, References and ReferenceNames would not be specified for an association and would be specified for elements that would be referenced by the association.

Table 9 provides an example of a typical "Operations" division based on option 3 in Table 7.

700 **Table 9 – "Operations" division example**

8.1 Profile conventions for operations

This profile defines intrinsic operations in terms of DSP0200.

For each profile class (including associations), the implementation requirements for operations, including those in the following default list, are specified in class-specific subclauses of this clause.

The default list of operations is as follows:

GetInstance( )

EnumerateInstances( )

EnumerateInstanceNames( )

For classes that are referenced by an association, the default list of operations includes the following operations in addition:

Associators( )

AssociatorNames( )

References( )

ReferenceNames( )

8.2 CIM_MemberOfCollection

Table 35 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless stated otherwise in Table 35, all operations in the default list in <pco-num> shall be implemented as defined in DSP0200.

NOTE: Related profiles may define additional requirements on operations for the profile class.

Table 35 – CIM_MemberOfCollection intrinsic operations

Operation	Requirements	Messages
CreateInstance( )	See 8.2.1	See Table 36
DeleteInstance( )	See 8.2.2	

Table 36 – CIM_MemberOfCollection.CreateInstance method standard messages

(return) MessageID	Message
(5) DMTF.CHK1	Invalid Transition State
(4) DMTF.CHK2	Operation not supported

8.2.2 CreateInstance( )

The CreateInstance( ) intrinsic operation may be implemented. If implemented, the creation of a MemberOfCollection instance associating a Zone instance representing a management zone with a Port instance representing a communication endpoint shall cause the addition of the communication endpoint into the management zone.

8.2.2 DeleteInstance( )

The DeleteInstance( ) intrinsic operation may be implemented. If applied to an instance of the MemberOfCollection association where the referenced member is a Port instance that represents a communication endpoint that is member of a management zone, the implementation shall remove that communication endpoint from the specified management zone.

6.5 Profile division "Use cases"

This division specifies use cases that demonstrate interesting behaviors or tasks provided by the profile.

A use case defines the interaction of an external actor and a server-side implementation in the execution of steps required to be performed in the realization of functionality described in the profile. This actor may be a CIM client or some other external entity (for example a person using a switch attached to the system). Use cases should represent a complete task from the perspective of the actor; this may involve multiple CIM operations or methods. A profile specification may document one or more use cases, each of which has different starting conditions and ending conditions. The purpose of the use case is to illustrate the steps required to accomplish some goal and the effects to the model in the course of accomplishing that goal.

The use cases may be presented as pseudo-code or free-form text. The use cases should include tasks that change the CIM elements or change the behavior of the instrumentation managed through the profile.

All extrinsic methods should be included in the use cases. A use case may include multiple (or no) extrinsic methods. A method may be included in multiple use cases. Detailed information about methods remains in the "Methods" division (see 6.4.1) and is not duplicated here.

Object diagrams or sequence diagrams should be included.

6.6 Profile division "CIM elements"

This division consists of:

- An overview subclause consisting of a table listing the profile's classes, indications, and queries
- A subclause for each class including a short description of the class and a table including the profile's use of properties and methods

The table (and a subclause) shall include all the classes and associations defined in the profile. It shall also include classes that are defined as part of other profiles and overridden (further constrained) by the current profile. In this case, only the overridden properties are included in the per-class subclause.

6.6.1 Overview subclause

This subclause shall be named "Overview" and shall contain a table listing the classes and indications of the profile. Table 10 is an example of this table.

Table 10 – CIM elements overview table example

Element	Requirement	Description
Classes		
CIM_StorageConfigurationService	Optional	
CIM_SystemDevice	Mandatory	
CIM_StorageCapabilities	Mandatory	
Indications		
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FCPort	Conditional	CQL, Conditional on support for the Indications Profile,
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_PortController	Optional	CQL
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_PortController	Optional	CQL

A table row representing a class shall include a reference to the division with detailed specification for that class.

For indications, the query language shall be included in the "Description" column. Indications should never be marked Mandatory; they may be Optional or Conditional on the Indications Profile.

The use of "Description" column or subclause of "Implementation" is specified in 5.5.

6.6.2 Supported classes, properties, and methods

Each class (including associations) supported by the profile shall have a separate subclause of the "CIM elements" division. The title of this subclause shall start with the class name. The subclause shall include one or more paragraphs describing the class and its relationship to the underlying implementation.

The subclause shall also include a table specifying all supported properties and methods as formatted in Table 11 if the profile describes expected behavior for the class. Expected behavior includes mandatory or conditional properties or methods, property formats and values, or text describing domain-relevant semantics. In other words, if this profile uses a class as described in the MOF, then the table of properties should be omitted. If the table is included, all key properties and properties with the “REQUIRED” qualifier shall be included in the table and marked mandatory. Other properties and methods may be included. If included, the "Requirement" column shall contain the word “Optional”, “Mandatory” or “Conditional”. If omitted, they are considered optional.

Table 11 – Class subclause example

10.7 CIM_IsSpare

IsSpare associates DiskDrives and RedundancySet. The DiskDrive may be automatically allocated for use if one of the members of RedundancySet fails.

Element	**Requirement**	**Constraints**
Antecedent	Mandatory	In this profile, shall be a DiskDrive
Dependent	Mandatory	
SpareStatus	Conditional	See 9.6
FailoverSupported	Conditional	See 9.7

Method names should be followed by “()”. The method signatures and return values shall not be documented here; instead, they are documented in the "Methods" division. A table row for a method should include a reference to the appropriate division of "Methods".

Optional properties and methods shall not be included in class tables unless there is some normative text that needs to be stated over and above what is in the MOF. The "Description" column or a subclause of the "Implementation" division shall specify normative requirements for each property. The normative information for each property shall include:

- If the property is conditional, the keyword “Conditional” shall appear in the "Requirements" column, the "Description" column shall contain either the condition itself or a reference to the subclause defining the conditional behavior (see 6.3).
- If the property is optional, the keyword “Optional” shall appear in the "Requirements" column.
- the property value pattern or constraint (see 6.6.2.1 and 6.6.2.2), if applicable
- a default value (see 6.6.2.3), if applicable
- cardinality (see 6.6.2.4)

- a description of behavior beyond that documented in the MOF may be included
- the keyword “Deprecated” if the property (or its use in the profile) is deprecated

The use of the "Description" column or subclause of the "Implementation" division is specified in 5.5.

Note that a class may appear multiple times in this division if the same class is used in different contexts. For example, a profile may use MemberOfCollection to associate members into different types of collections. A short title for the context, in parenthesis, should follow the class name in the subclause heading. For example:

> 10.7 CIM_StoragePool (Concrete Pools)
>
> …
>
> 10.8 CIM_StoragePool (Primordial Pools)

The explanatory paragraphs in this subclause then describe the different contexts.

6.6.2.1 Specifying property value patterns for strings

In string properties, it is possible to represent the same value with different formats. For example, the same binary value may be represented in hexadecimal or decimal; the hexadecimal representation may have a “0x” prefix or “h” suffix, and the value may include dot, space, or colon separators. Although these are valid alternative formats, it is a challenge for client applications to determine which formats are being used and to determine whether two values are equivalent.

To assure client interoperability, profile instrumentation shall specify a mandatory format for all properties. Formats shall be specified in both a normative text description and a regular expression. The regular expression may be used in code that validates formats, but may not be intuitive to all profile readers. The keyword “pattern” shall be used to identify the regular expression.

Example

> :PermanantAddress shall be formatted as 16 unseparated uppercase hex digits (pattern "^[0123456789ABCDEF]{16}$")

The regular expression syntax is defined in Annex C.

In some cases, valid formats for one property are specified in another property. For example, ComputerSystem.NameFormat is an enumeration with values that imply formats for ComputerSystem.Name. A profile specification should provide normative text and regular expressions for each valid format.

Example – the normative text for Name:

> If NameFormat matches “IP”, Name represents an IP address and shall be formatted as specified in 5.3.2.5.
>
> If NameFormat matches “HID”, Name represents a hardware ID as specified in [T10 SPC] and shall be formatted as 16 unseparated uppercase hex digits (pattern "^[0123456789ABCDEF]{16}$")
>
> If NameFormat matches “WWN”, Name represents a Fibre Channel WWN and shall be formatted as 8 unseparated uppercase hex digits (pattern "^[0123456789ABCDEF]{8}$")

The normative text for NameFormat should then specify that IP, HID, and WWN are the only valid values (see 5.4).

6.6.2.2 Specifying property value constraints

In some cases, a profile may constrain a property to a single value or a set of values. The formats described in this subclause are a subset of regular expressions (see Annex C) and shall be used to specify value constraints of properties. Properties of any type (string, integer, …) may be constrained in a profile specification. If a string property is constrained to particular values, a regular expression pattern (see 6.6.2.1) shall not be specified.

For properties without a values qualifier, a single valid value shall be specified as the name of the property followed by "matches" followed by the value. For example:

BlockSize matches 512[1]

For properties without a values qualifier, a list of valid values shall be specified as the name of the property followed by "matches" followed by a list of values separated by vertical bars. For example:

BlockSize matches 512|520

For properties with a values qualifier, a single valid value shall be specified as the name of the property followed by "matches" followed by the valuemap followed by an open parenthesis followed by the value followed by a close parenthesis. For example:

ProtocolIFType matches 4096 (IP v4)

For properties with a values qualifier, a list of valid values shall be specified as the name of the property followed by "matches" followed by a list of values separated by vertical bars followed by an open parenthesis followed by a list of values (separated by “ or “) followed by a close parenthesis. For example:

ProtocolIFType matches 4096|4097|4098 (IP v4 or IP v6 or both)

Note that the lists of valuemaps and values are separate. This allows the valuemap list to be a valid regular expression. This approach enables automatic generation of profile specification tables from a separate source (such as XML) that can also be used for testing. If the valuemaps and values were mixed [for example, ProtocolIFType matches 4096 (IP v4) | 4097 (IP v6), | 4098 (both)], the result is not a valid regular expression.

In all cases, <property-name> may be omitted when the pattern description is included in a table in the "CIM elements" division.

A profile specification may constrain the possible values (or valuemaps) from MOF, but shall not extend them. For example, a profile specification may specify a higher minimum value compared to the MINVAL in the MOF.

6.6.2.3 Specifying default property values

A profile specification may specify a default value for a property. The default value is the value server-side instrumentation shall return unless overridden. This appears as "Default value is" followed by the value. The default value shall comply with any property values (see 6.6.2.2) as well as minimum or maximum values from the MOF.

[1] This terse example (and others that follow) apply when the value constraint is expressed in a table cell. In a paragraph, the author may opt to work the format (after “matches” in the other examples) into sentences. For example:

BlockSize holds the formatted block or sector size and shall have the value 512 for conformance to this profile.

6.6.2.4 Specifying Cardinality

Reference properties in association classes shall include text specifying the cardinality if the MOF cardinality if overridden by the profile. The format is "Cardinality" followed by the cardinality. The cardinality may be a single value or a pair of values separated by two periods. A value may be:

1	indicating 1 and only 1 reference
*	indicating 1 or more references
m..*n*	where *m* is 0 or a positive integer and *n* is a positive integer or * representing 1 or more

It is also valid to say “Cardinality conforms to the MOF.”

If no cardinality is specified, the cardinality from the MOF applies to the profile.

Annex A
(informative)

Relationships between profile elements

Instances of classes in one profile may have associations to instances of classes in other profiles. The most common type of cross-profile association is scoping. Most CIM classes have a scoping association (for example, SystemDevice, HostedService, HostedAccessPoint) to a ComputerSystem. In general, the ComputerSystem instance is in a profile and the scoped instances are in either the profile or its associated component profiles.

Figure 1 depicts a common configuration of an autonomous profile and two component profiles. The ComputerSystem and class A are elements of the AP1 Profile. Component profile CP1 has classes B, C, and D. Component profile CP2 has classes E and F.

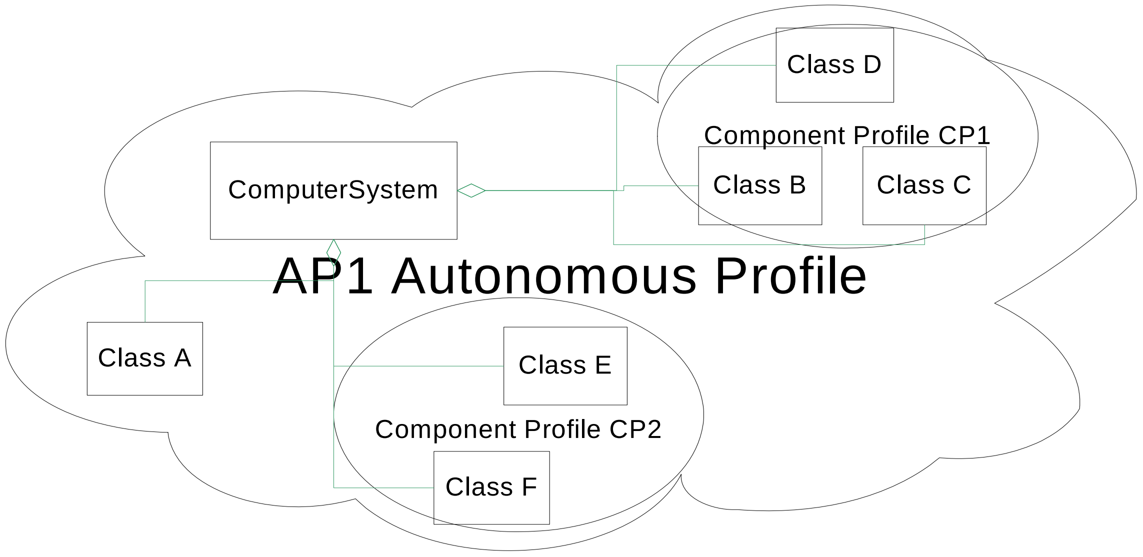


Figure A-1 – Relationships between elements

The aggregation associations represent scoping associations and appear to float between the ComputerSystem in the autonomous profile and elements of the component profiles rather than being part of either. This specification requires that these associations be specified as part of the component profiles (CP1 and CP2 in this example).

Registration of profiles provides a method that allows a server-side implementation to inform clients whether profiles — and the scoping associations to profile elements — are supported.

Annex B
(normative)

Structure of a DMTF profile specification

A profile specification published by DMTF shall follow the DMTF specification standards and shall include all profile divisions listed in Table 1 as clauses, resulting in the document structure shown in the frame below.

DSP1000 (*Management Profile Specification Template*) is an informational document that may be used as a template for profile specifications published by DMTF.

Foreword

Introduction (optional)

1. Scope
2. Conformance (optional)
3. Normative References
   - shall include a reference to the CIM Infrastructure Specification
   - shall include a reference to the appropriate operations specification (CIM operations over HTTP; later replaced by the appropriate operations mapping specification)
   - shall include a reference to the appropriate specification including the definition of profile regular expressions (initially this is the Profile Usage Guide)
   - shall include a reference to the DMTF specification that explains keywords as defined in clause 3
4. Terms and Definitions
   - shall include terms this usage guide identifies as keywords (may, shall, …)
5. Symbols and Abbreviated Terms (optional)
6. Synopsis
7. Description
8. Implementation
9. Methods
10. Use Cases
11. CIM Elements

Annexes (optional)

Change Log (optional in preliminary versions, removed when published)

Bibliography (optional)

Annex C
(normative)

Regular expression syntax

This annex defines the regular expression syntax used in profile specifications to specify the format of values, especially those representing identifiers. The regular expression grammar below uses Augmented BNF (ABNF) as defined in RFC5234 with the following exceptions. Rules separated by a bar (|) represent choice (Instead of using a slash (/) as defined in ABNF). Ranges of alphabetic characters or numeric values are specified using two periods (..) placed between the beginning and ending values of the range (instead of using the minus sign (-), as defined in ABNF).

The rules defined in this syntax are assembled into a complete query by assuming white space characters between them, except where noted otherwise. (ABNF requires explicit specification of white space.)

The comma (,) is used to explicitly designate concatenation of rules (instead of implicit concatenation of rules as specified by ABNF).

Note:

1) ABNF is NOT case-sensitive.
2) The rules above apply to the ABNF used here and NOT to the resultant Regular Expression used in Full or Basic Like. In particular, except where noted, white space is significant within the resultant Regular Expression.
3) Reference to UNICODE-CHAR refers to UNICODE-CHAR as defined in RFC3629.

The regular expression syntax defined in this annex is a subset of the POSIX Extended Regular Expression BNF defined in POSIX Regular Expressions.

```
pre-special-char = "." | "\" | "[" | "^" | "$" | "*" | "+" | "?" | "|"
```

Special Characters
'.' matches any single character
'\' escapes the next character so that it isn’t special
'[' starts a bracket expression
"^" when used as a left-anchor
"$" when used as a right-anchor
'*' 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.
‘|’ represents a choice

```
pre-ordinary-char= UNICODE-CHAR
```

A character, other than a pre-special-char

```
pre-escaped-char = "\." | "\\" | "\[" | "\^" | "\$" | "\*" | "\+" | "\?" | "\|"
```

escaped special char

```
pre-bracket-char = "[", *(pre-ordinary-char | pre-escaped-char), "]"
```

Square brackets ('[' and ']') are used to enclose characters, any one of which may be matched. For example, 'r[au]t' matches 'rat' or 'rut'.

A "]" can be added to the set by making it the first character in the set.

To match any character except what is specified in the square brackets, follow the opening bracket with a caret ('^')

```
pre-single-char = "." | pre-ordinary-char | pre-escaped-char | pre-bracket-char
```

Single character regular expression

```
pre-multi-char = pre-single-char,"*"
```

Matches multiple occurrences of a single character

```
pre-dup-symbol = "*" | "+" | "?"
        | "{", pre-unsigned-integer, [",",  [pre-unsigned-integer]], "}"
```

'*' indicates that the preceding item is matched zero or more times.

'.*' combines the first two special characters together to indicate that any sequence of characters is matched.

'?' indicates that the preceding item is optional and will be matched at most once.

'+' indicates that the preceding item will be matched one or more times.

```
pre-expression = pre-single-char
        | "^"
```

To force a match at the beginning of a string, start the character string with '^'. Note that "^]" is used to include "]" as the first character of the string.

```
        |  "$"
```

To force a match at the end of a string, end the character string with '$'.

```
        | "(", pre-multi-char, ")"
```

Parentheses can be used to define the order of evaluation.

```
        | pre-expression, pre-dup-symbol
```

```
pre-branch = [pre-branch], pre-expression
```

```
pre-extended-reg-exp = [pre-extended-reg-exp, "|"], pre-branch
```

Profile regular expression: To represent a choice, use the vertical bar character (|).

Annex D
(normative)

Profile Specialization

In some cases, multiple profiles may include substantial overlap. One type of overlap is a profile that extends the capabilities of another. For example, a profile for sets of batteries supporting failover may be based on a generic battery profile. Another possible overlap is a set of profiles that specialize a common abstract profile. For example, a common profile for the generic aspects of a TCP service, with specialized profiles for specific services. The primary advantages in using profile specialization are avoiding duplicate documentation and avoiding inadvertent differences when similar profiles fall out of synchronization.

There is some similarity between profile specialization and inheritance — the behavior in a specialized profile inherits the behavior from the profiles it specializes. In some cases, specialized profiles may specify subclasses of classes from their abstract profiles. However, there are some major differences as well. There is no compiler or CIMOM awareness of profile specialization; we want to assure that clients can successfully see generic profiles in all specializations, but this is only enforced by the diligence of the profile specification authors.

D.1 Normative Requirements for Profile Specialization

For the purpose of this annex, “abstract profile” refers to the generic profile that is specialized by a “specialized profile”.

A profile may specialize one or multiple abstract profiles.

A profile specifies requirements for its classes, indications, and other profiles. In addition, a profile specifies its requirements for properties and methods of classes. The requirements of elements are defined in the following taxonomy — “not specified”, “optional”, “conditional”, and “mandatory”. This is an ordered list – from least constrained (not specified) to most constrained (mandatory). A specialized profile may make a requirement more constrained than the requirements in its abstract profile (that is, an optional property in the abstract profile may be redefined as mandatory in a specialized profile). Requirements shall not be reduced in a specialized profile (that is, a mandatory property in the abstract profile shall not be redefined as optional in a specialized profile).

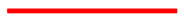
A specialized profile may replace a class from an abstract profile with one of its subclasses. A specialized profile shall not replace a class from an abstract profile with a class that is not a subclass of the class from the abstract profile. A subclass specified in a specialized profile inherits all the constraints of the superclass in the parent profile. The value of a property in a specialized profile may be constrained in a specialized profile. For example, if an abstract profile allows a property to have the values “4”, “5”, or “6”, the specialized profile may limit this property to “4” or ”5”. A specialized profile shall not introduce additional values beyond those defined in its abstract profiles.

Annex E
(informative)

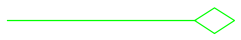
Diagram Conventions

At the time this guide is being written, DMTF has no specific document for diagram conventions. All the schema diagrams associated with MOFs follow conventions described in a legend in each diagram. The subset of conventions that apply to diagrams in profile specifications are described here:

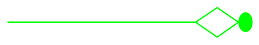
Associations – red line



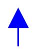
Aggregation relations – green line with a diamond at one end



Composition Aggregation relations – green line with a diamond and a dot at one end



Inheritance relationships – blue line with arrow at the superclass end:



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

{D}

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

{E}

DMTF is considering other diagram conventions, in particular, UML compliant diagrams. Diagrams in new profile specifications should comply with the diagram conventions that DMTF adopts.

Annex F (normative)

Experimental Content

A profile specification may include experimental content. Experimental content is informational. Experimental content may include normative language. Experimental content is provided as information on the direction and current thinking of the authors of a profile specification.

If experimental content is included in a profile specification it shall be formatted as follows:

- If experimental content is in a table, the table shall include a "Comments" or "Description" column and the word EXPERIMENTAL in capital letters shall be at the beginning of the text in the column.
- If the experimental content is text (not in a table), the start of the experimental content shall begin with a double line the width of the page followed with the EXPERIMENTAL word in capital letters starting at the beginning of a line and on the line by itself. The end of the experimental content shall end with the EXPERIMENTAL word in capital letters starting at the beginning of a line and on the line by itself followed by a double line the width of the page.

Experimental content shall not span subclauses.

Table example:

Table x – <profile>: CIM Elements

Element	Requirement	Description
CIM_ProtocolEndpoint	Mandatory	See …
CIM_IPProtocolEndpoint	Optional	
CIM_ATAProtocolEndpoint	Mandatory	EXPERIMENTAL

Text example:

EXPERIMENTAL

Experimental content …

EXPERIMENTAL

Annex G
(Informative)

Change Log

Version	Date	Description
1.0.0	2006-06-14	Release as DMTF Final Standard
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

Bibliography

UML Specifications
http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML

UML Intro: *Practical UML^{TM}, A Hands-On Introduction for Developers*
http://bdn.borland.com/article/0,1410,31863,00.html

DMTF DSP1000, *Management Profile Specification Template 1.0,*
http://www.dmtf.org/standards/published_documents/DSP1000_1.0.pdf