**Document Number: DSP1102**

**Date: 2010-10-21**

**Version: 1.0.0**

# Launch in Context Profile

**Document Type: Specification**

**Document Status: DMTF Standard**

**Document Language: en-US**

# CONTENTS

# Figures



# Tables

161

# Foreword

The Launch in Context Profile (DSP1102) was prepared by the Physical Platform Profiles Working Group of the DMTF.

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

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

Author:

- George Ericson – EMC

Contributors:

- Martine Wedlake – IBM
- John Crandall – Brocade

Reviewers from the SNIA Core Working Group:

- Duane Baldwin – IBM
- Paul Von Behren – Symantec
- Steve Peters – HP

Reviewers from the DMTF Physical Platform Profiles Working Group:

- Michael Johanssen – IBM
- Larry Lamers – VMware

# Introduction

The information in this specification should be sufficient for a provider or consumer of this data to identify unambiguously the classes, properties, operations, methods, and values that shall be instantiated and manipulated to represent and manage one or more launch points. Each launch point provides information required to launch a management application in the context of the managed system.

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

## Document conventions

### Typographical conventions

The following typographical conventions are used in this document:

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

### Deprecated material

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

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

The following typographical convention indicates deprecated material:

---

**DEPRECATED**

Deprecated material appears here.

**DEPRECATED**

---

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

## 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.

# Launch in Context Profile

## 1 Scope

This profile specifies common management interfaces for the purpose of managing and discovering information about auxiliary services that are available for particular resources or classes of resources represented by instances of CIM_ManagedElement. This information includes the address of the each auxiliary service, the features supported by that service, the contextual information available for that service, and the capabilities for customizing that information.

## 2 Normative references

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

DMTF *CIM Schema*, 2.27,
http://www.dmtf.org/standards/cim

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

DMTF DSP0207, *WBEM URI Mapping 1.0*,
http://www.dmtf.org/standards/published_documents/DSP0207_1.0.pdf

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

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

DSP1001 v1.1 is not yet approved:
Publication of this specification is dependent on approval.

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

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

IETF RFC3986, *Uniform Resource Identifier (URI): Generic Syntax*, Jan. 2005,
http://www.ietf.org/rfc/rfc3986.txt

IETF RFC5234, *ABNF: Augmented BNF for Syntax Specifications, January 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/4230517/ISO_IEC_Directives__Part_2__Rules_for_the_structure_and_drafting_of_International_Standards__2004__5th_edition___pdf_format_.pdf?func=doc.Fetch&nodeid=4230517

*W3C, XQuery 1.0 and XPath 2.0 Functions and Operators,* 23 January 2007, http://www.w3.org/TR/xpath-functions/

W3C, *URIs, Addressability, and the use of HTTP GET and POST*, TAG Finding 21 March 2004, http://www.w3.org/2001/tag/doc/whenToUseGet-20040321

# 3 Terms and definitions

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

The terms "shall" ("required"), "shall not," "should" ("recommended"), "should not" ("not recommended"), "may," "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Annex H. The terms in parenthesis are alternatives for the preceding term, for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that ISO/IEC Directives, 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 document, clauses, subclauses, or annexes labeled "(informative)" do not contain normative content. Notes and examples are always informative elements.

The terms defined in DSP0004, DSP0223, DSP1001, and DSP1033 apply to this document. The following additional terms are used in this document.

**3.1**
**launch in context**
the act of invoking a service with contextual information

**3.2**
**launch point**
information within a managed system that provides information about a service that can be invoked

# 4 Symbols and abbreviated terms

The abbreviations defined in DSP0004, DSP0223, DSP1001, and DSP1033 apply to this document. In addition, the following abbreviations also apply.

**4.1**
**HDR**
host discovered resources

**4.2**
**LUN**
logical unit number

**4.3**
**SCSI**
small computer systems interface

**4.4**
**VPD**
vendor product definition

# 5 Synopsis

**Profile Name:** Launch in Context

**Version:** 1.0.0

**Organization:** Physical Platform Profiles Working Group

**Abstract indicator**: false

**Profile type**: component

**Schema**: DMTF CIM 2.27

**Central class adaptation**: LaunchService

**Scoping class adaptation**: PublishingSystem

**Scoping algorithm**: HostedService

## 5.1 Summary

This profile provides discovery and management of launch-in-context information. This enables a primary management service (for example, the CIM Service supporting this profile), to provide its clients sufficient information to launch known auxiliary management services. Management clients may utilize this information to provide end users the ability to launch auxiliary management services in context.

## 5.2 Related Profiles

An implementation that is conformant to this profile shall also be conformant with the requirements of the implemented related profiles.

Table 1 identifies profiles that are referenced by this profile.

**Table 1 – Related profiles**

| Profile Name | Organization | Version | Requirement | Description |
|---|---|---|---|---|
| Profile Registration | DMTF | 1.0 | Mandatory | Specifies rules utilized to advertise support for this profile. |
| Indications | DMTF | 1.1 | Conditional | Specifies rules that enable clients to discover and subscribe to indications supported by this profile. The conformance requirement is conditional on the LaunchPointMonitoring feature (see 7.1.3). |

## 5.3 Features

Table 2 lists the features described in this profile.

326 **Table 2 – Features**

| Feature name | Granularity | Requirement | Description |
|---|---|---|---|
| ScopedLaunchPoints | Profile | Optional | See 7.1.1. |
| ClientPublishing | Profile | Optional | See 7.1.2. |
| LaunchPointMonitoring | Profile | Optional | See 7.1.3. |
| ParameterDerivation | Profile | Optional | See 7.1.4. |

327 ## 5.4 Events

328 Table 3 lists the events specified by this profile. See the LifecycleFilter (7.2.14), FilterCollection (7.2.3),
329 and FilterCollectionMember (7.2.4) adaptations for the implementation requirements necessary to detect
330 each event.

331 **Table 3 – Events**

| Name | Description |
|---|---|
| DMTF:LaunchPoint:Added | Indicates that a LaunchPoint instance has been added |
| DMTF:LaunchPoint:Deleted | Indicates that a LaunchPoint instance adaptation has been deleted |
| DMTF:LaunchPoint:Modified | Indicates that a LaunchPoint instance has been modified |

332 ## 5.5 Adaptations

333 Table 4 identifies the class adaptations defined in this profile.

334 **Table 4 – Adaptations**

| Adaptation | Elements | Requirement | Description |
|---|---|---|---|
| **Classes** | | | |
| CurrentCapabilities | CIM_ElementCapabilities | Conditional | See 7.2.2. |
| HostedLaunchPoint | CIM_HostedAccessPoint | Mandatory | See 7.2.5. |
| HostedLaunchService | CIM_HostedService | Mandatory | See 7.2.6. |
| FilterCollection | CIM_FilterCollection | Conditional | See 7.2.3. |
| | IndicationsProfile::FilterCollection | | |
| FilterCollectionMember | CIM_MemberOfCollection | Conditional | See 7.2.4. |
| | IndicationsProfile::MemberOfCollection | | |
| LaunchCapabilities | CIM_LaunchInContextCapabilities | Conditional | See 7.2.7. |
| LaunchPoint | CIM_LaunchInContextSAP | Mandatory | See 7.2.8. |
| LaunchProfile | CIM_RegisteredProfile | Mandatory | See 7.2.12. |
| | ProfileRegistrationProfile::Implementation | | |
| LaunchService | CIM_LaunchInContextService | Mandatory | See 7.2.13. |
| | ProfileRegistrationProfile::CentralElement | | |
| LifecycleFilter | CIM_IndicationFilter | Conditional | See 7.2.14. |
| | IndicationsProfile::IndicationFilter | | |
| ManagesLaunchPoint | CIM_ServiceAffectsElement | Mandatory | See 7.2.15. |
| PublishingSystem | CIM_System | Mandatory | See 7.2.16. |

| Adaptation | Elements | Requirement | Description |
|---|---|---|---|
| | ProfileRegistrationProfile::ScopingElementIndicationsProfile::IndicatingSystem | | |
| ScopedElement | CIM_ManagedElement | Conditional | See 7.2.17. |
| ScopesElement | CIM_ManagementSAP | Conditional | See 7.2.18. |
| ScopingProfile | CIM_RegisteredProfile | Mandatory | See 7.2.19. |
| **Indications** | | | |
| LaunchPointAdded | CIM_InstCreation | Conditional | See 7.2.9. |
| | IndicationsProfile::InstCreation | | |
| LaunchPointDeleted | CIM_InstDeletion | Conditional | See 7.2.10. |
| | IndicationsProfile::InstDeletion | | |
| LaunchPointModified | CIM_InstModification | Conditional | See 7.2.11. |
| | IndicationsProfile::InstModification | | |

## 5.6 Use cases

Table 5 lists the use cases described in this document.

**Table 5 – Use cases**

| Use case | Description |
|---|---|
| PublishSpecifiedLaunchPoint | Application causes a specified launch point to be published.<br>See 8.1.2. |
| FederateLaunchPoints | A management system aggregates launch points from other CIM-based management systems.<br>See 8.1.3. |
| PublishDiscoveredLaunchPoints | A management system publishes launch points from underlying, non-CIM subsystems.<br>See 8.1.4. |
| ProfileSpecifiedLaunchPoints | A management profile may specify launch points to be implemented in a conformant system.<br>See 8.1.5. |
| DiscoverConformance | Determine if the managed system advertises support for this profile.<br>See 8.2.1. |
| ListLaunchServices | Find LaunchService instances that are conformant to this profile.<br>See 8.2.2. |
| ListLaunchPoints | List advertised LaunchPoints.<br>See 8.2.3. |
| AddLaunchPoint | A management client adds a new LaunchPoint to the PublishingSystem.<br>See 8.2.4. |
| RemoveLaunchPoint | A management client removes a LaunchPoint from the PublishingSystem.<br>See 8.2.5. |
| GetDerivedParameters | Ask the PublishingSystem to return context specific values for the parameters of a LaunchPoint-specified URI template.<br>See 8.2.6. |

# 6 Description

This profile provides discovery and management of launch-in-context information. This enables a primary management service to provide its end users the ability to launch auxiliary management services in context.

The services described by this profile assume several active entities:

- an end user, or an administrator
- a common user interface (browser-based, Java application, or other)
- a primary management service (or back-end management server), which might manage a single system or multiple systems
- one or more publishing systems (interfaced through a WBEM protocol)
- any number of auxiliary management services (typically an element or device manager) that provides additional management services for the resources known to publishing systems
- any number of other managed systems, where "other" is a relative term meant to imply that the system or interface may be different from that of a publishing system. In practice, an "other" system, as shown in Figure 1, might be a publishing system. Similarly, the resource known to the publishing system may be the same managed element that is known to the other managed system; however, it is more likely that the management representation of the referenced resource is not shared across the two systems.

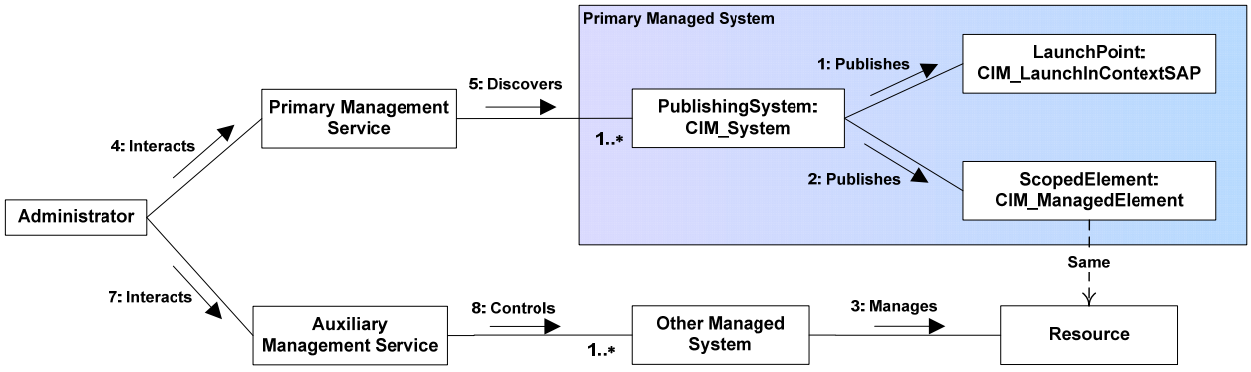



**Figure 1 – Launch in Context overview diagram**

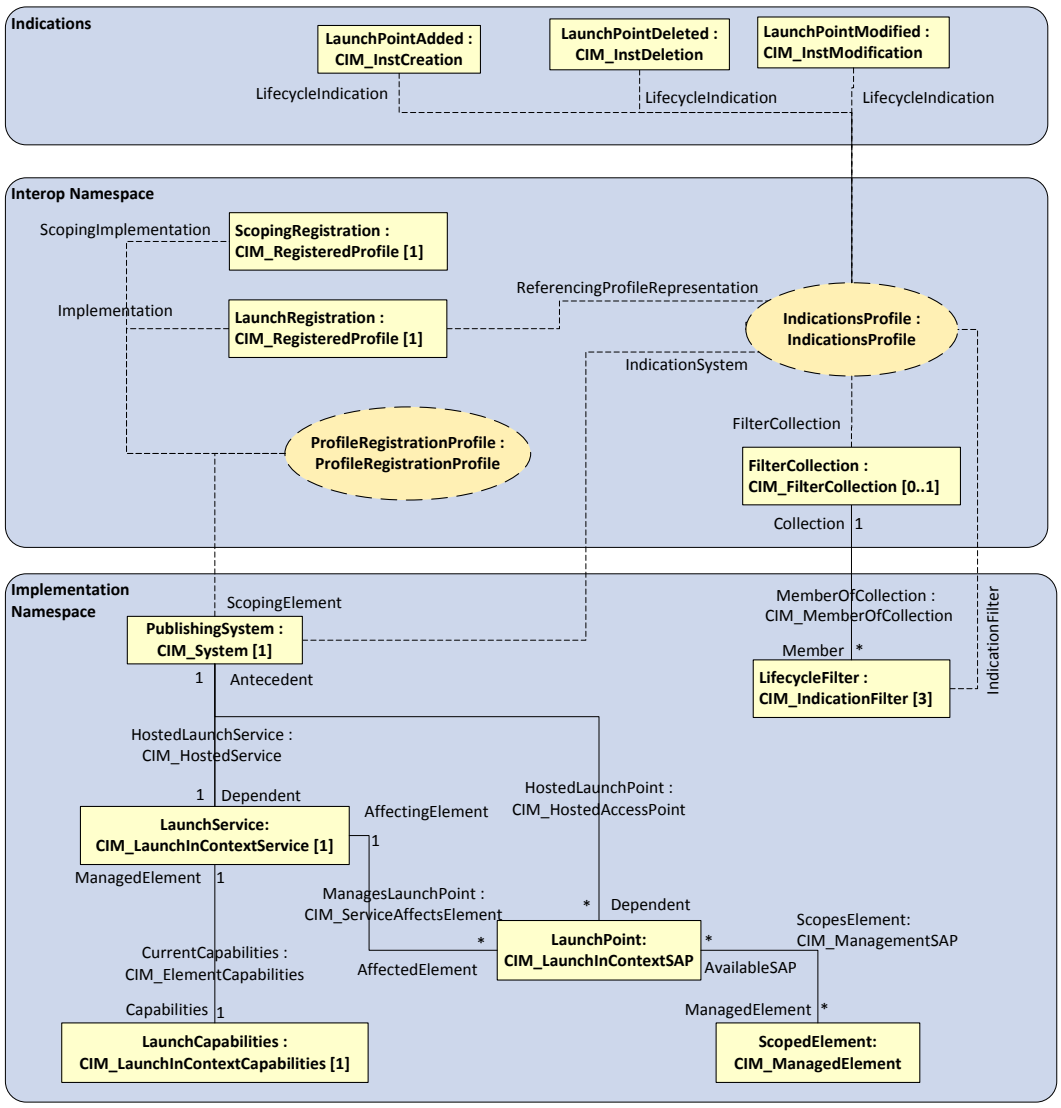



**Figure 2 – Overview collaboration structure diagram**

## 6.1 Publishing system

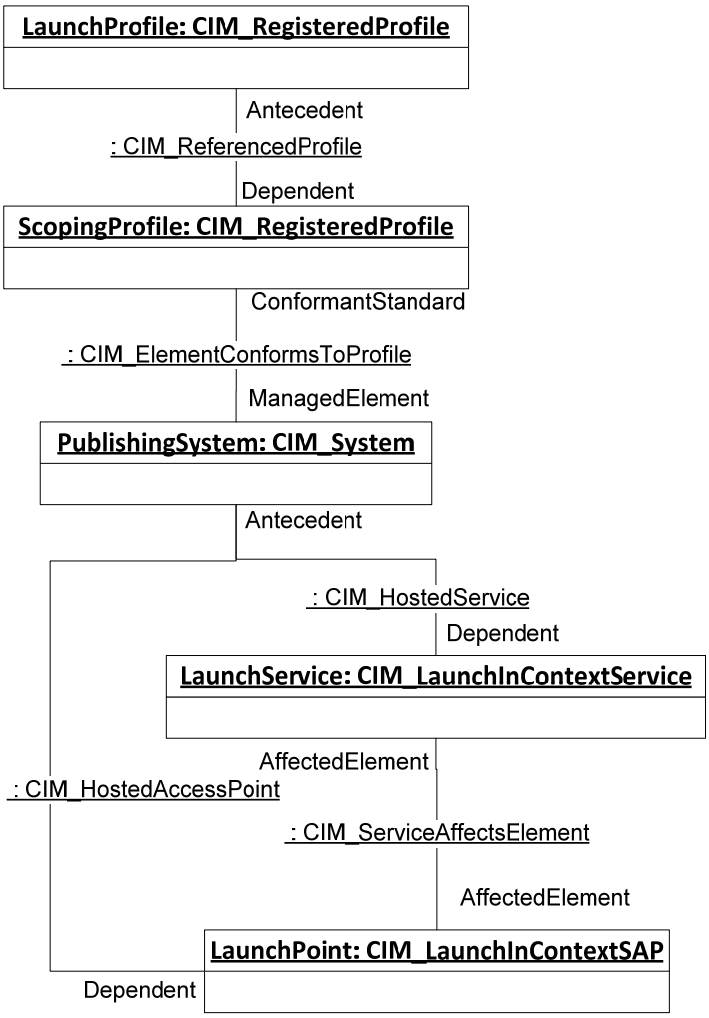
As shown in Figure 1, the publishing system makes launch points available (represented by an instance of CIM_LaunchInContextSAP). Each launch point contains a URL to an auxiliary service that provides additional management capability for various instances in the CIM Namespace of the publishing system. Those instances represent resources that may be part of the publishing system or, more typically, have been discovered by the publishing system but are actually part of some other system.

This profile assumes either of the following situations:

- The vendor that implements a publishing system has prior knowledge of one or more auxiliary services.
- Some management service has knowledge of one or more auxiliary services.

The latter case is supported by the ClientPublishing feature (see 6.5.2).

372 Figure 3 shows the minimum set of instances instantiated by the Publishing System to advertise a single
373 conformant LaunchPoint. The example assumes a scoping profile, such as SNIA's Array, Switch, or
374 Fabric profiles or such as the Base Server profile (DSP1004). The PublishingSystem would also be the
375 central instance of that scoping profile.




376

377 **Figure 3 – Minimal instantiation**

## 378 6.2 Launch Point

379 At its simplest, the launch-in-context information includes:

380 • the address of the auxiliary service

381 • information that enables the user of the management client to decide whether to launch the
382 auxiliary service

383 • contextual information to pass on invocation

384 In addition to the address of an auxiliary service, each launch point defines a set of classes. Only
385 instances that belong to this set are expected to be managed by the auxiliary service.

Also, each launch point specifies a set of parameters that are used by the management client to modify the URL of the auxiliary service to provide context information.

This information enables the administrator together with its primary management service to select and launch an auxiliary service with context information. The auxiliary service is invoked by doing an HTTP GET to the updated URL. The W3C TAG Finding, *URIs, Addressability, and the use of HTTP GET and POST*, states:

> *"In particular, the convention has been established that the GET and HEAD methods SHOULD NOT have the significance of taking an action other than retrieval. These methods ought to be considered "safe". This allows user agents to represent other methods, such as POST, PUT and DELETE, in a special way, so that the user is made aware of the fact that a possibly unsafe action is being requested."*

In the context of launching management applications, this recommendation can be interpreted to mean that the application itself should not be changed. Therefore, the proposed usage of GET is consistent with this recommendation for most cases.

## 6.3 Primary Management Service

The primary management service is a client application that understands this profile and interfaces with the publishing system. The primary role defined by this profile is to present information recorded in published launch points about auxiliary management services, and then to provide the ability to launch one of those auxiliary management services.

The goal of this profile is to enable an administrator together with its primary management service to discover enough information about these auxiliary management services to allow them to be launched with contextual information. This ability is known as "Launch in Context".

The mechanism to launch services provided by this profile is not limited to launching management applications. Any Web-based service can be referenced. In some cases, those services do change the state of the addressed object. For consistency with the W3C recommendation, such services should be invoked with the appropriate HTTP operation.

Context information for a launch is provided by first specifying the URL as a parameterized template. The primary management service must replace the parameters in the template with parameter values to derive a URL containing context information.

The publisher of the launch point may specify operations other than HTTP GET, for example, HTTP PUT. Such operations typically include a message as part of the operation. A parameterized launch message may be published with the protocol operation necessary to enable the primary management service to send those messages to initiate a launched service.

This profile assumes that the primary management service understands each parameter name and is able to derive correct parameter values from information it holds, from the user of the primary management service or from the managed system. This understanding is based on the parameter name. For this reason, each well-known parameter should be formed with the defining organization and specification as part of its name.

This profile also allows additional parameter value validation rules to be published by the launch point. If it is able, the primary management service may use these rules to validate parameter values that it solicits from an end user, before those values are passed to the launched service. This facility is an aid for a generic management service to use in the process of obtaining valid parameter values required to launch an auxiliary service.

## 6.4 Auxiliary Management Service

The operational assumption reflected in the management model is that the auxiliary management service is remote to the publishing system. However, the auxiliary service may be hosted on the primary system without the need for a different model.

The auxiliary management services are expected to be network accessible to the administrator. The operational presumption is that those auxiliary services themselves do not have access to the publishing system; however, it is often the case that they do. (In fact, the auxiliary management service may directly access the publishing system.) There is also no requirement that the managed system understands or has access to any of the auxiliary services.

## 6.5 Optional Profile Features

The optional features of the profile are as follows:

- Launch point scoping, an optional interface to support restricting the set of resources to which the launch information applies (see 6.5.1)
- Client published launch points, an optional interface to support the ability for the primary management service to publish its knowledge of auxiliary services (see 6.5.2)
- LaunchPointMonitoring, an optional capability to efficiently monitor the published launch information (see 6.5.3)
- ParameterDerivation, advanced launch point context value derivation support, an optional feature that provides automation of the derivation of contextual information required for a launch (see 6.5.4)

### 6.5.1 ScopedLaunchPoints feature

This optional feature enables a particular launch point (represented by an instance of CIM_LaunchInContextSAP) to be constrained to a list of instances.

If not constrained, then a launch point applies to all instances of the classes listed in the ManagedClasses property. This puts the burden on the auxiliary management service to filter out and reject requests on resources it does not manage. Depending on use, this may or may not be acceptable. This feature moves the burden of resource filtering to the primary management service and its managed systems.

An alternative filtering option is to use a more restrictive subclass; however, that option is not always appropriate, for example in cases where property values or relationships to other instances are the discriminators.

The provisions that specify this optional feature are described in 7.1.1.

### 6.5.2 ClientPublishing feature

The typical enterprise environment is dynamically changing, sometimes at evolutionary speeds and sometimes much more rapidly.

The optional ClientPublishing feature provides the interfaces required for the primary management service to install launch points into the publishing system.

With the base capability, launch points are added with new versions of, or through extensions to, the implementations of the publishing system. This requires either support through either of the following:

- The vendor implementing the managed system to have prior knowledge of all auxiliary services and to create appropriate launch points to represent them

- The managed system vendor to support installation of third-party providers that contain knowledge of auxiliary services and those providers create appropriate launch points to represent those services

This feature assumes that either the administrator or some management service is aware of the auxiliary services in the enterprise and has responsibility for the installation and configuration of those services.

NOTE: Launch points are not typically used directly by the managed system in which they are installed. Rather, they enable management services to discover and utilize those launch points to launch the auxiliary services they describe.

The provisions that specify this optional feature are described in 7.1.2.

### 6.5.3 LaunchPointMonitoring feature

Management services may choose to do a global discovery of the launch points within a managed system. Having done that, they then need a means to know if the topology of launch points changes in that system. One approach is simply to enumerate again and compare the results. However, this can be expensive in a large system. Additionally, that simple algorithm is not sufficient if the optional scoping feature is supported.

The optional LaunchPointMonitoring feature supports the ability for the primary management service to subscribe to changes in the launch point topology within a managed system.

The provisions that specify this optional feature are described in 7.1.3.

Figure 4 shows an example set of instances needed to implement the LaunchPointMonitoring feature.

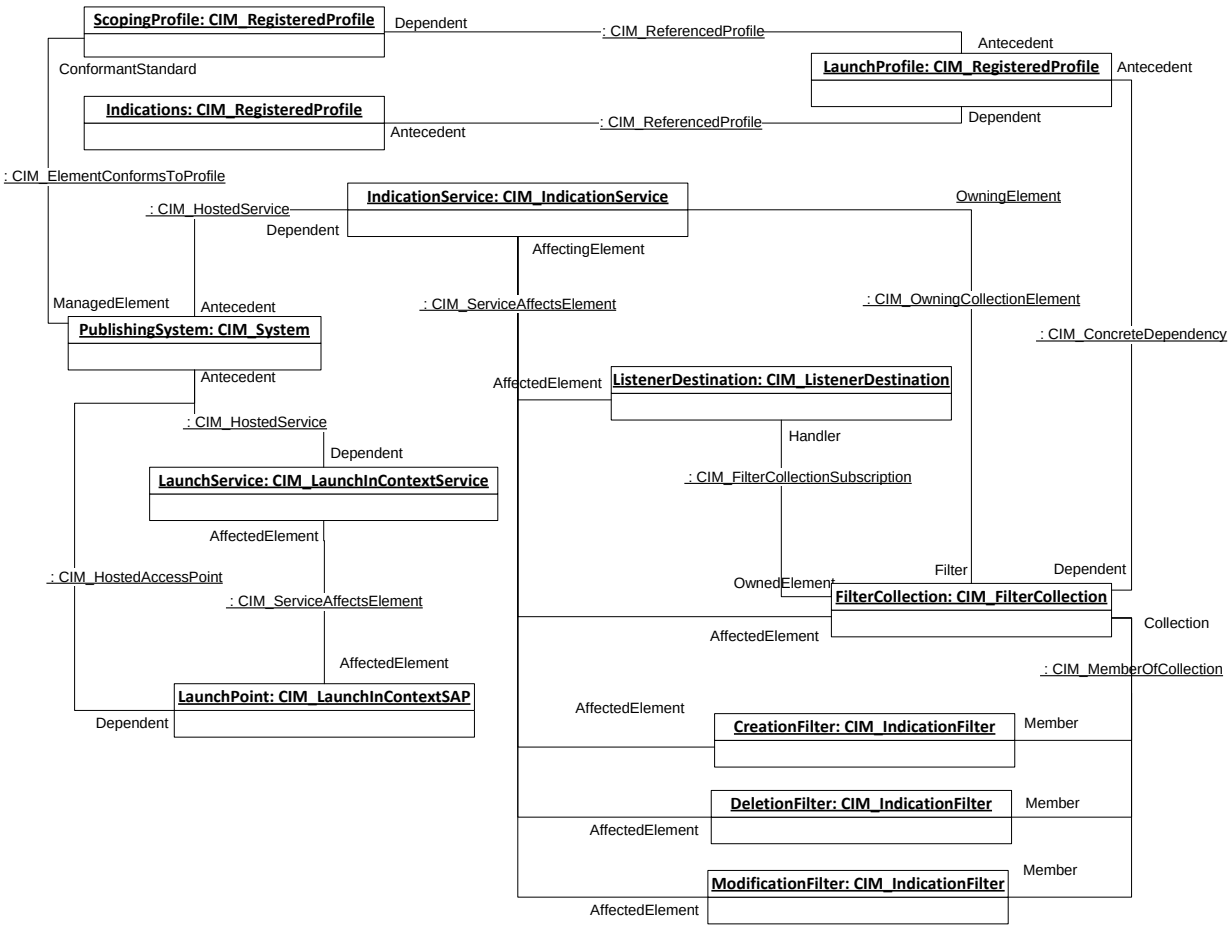



**Figure 4 – Example instantiation of LaunchPointMonitoring feature**

### 6.5.4 ParameterDerivation feature

With only the basic features of the profile, the primary management service must know how to derive values for each context parameter as a function of only the parameter's name. This is sufficient if the environment is limited to a set of well-known profiles and services. However, the enterprise environment could be evolving faster than the primary management service can keep up (or the speed at which the customer is willing to update it). In that dynamic environment, there needs to be a means for the primary management service to provide values for parameters without derivation being dependent on the parameter's name.

The optional ParameterDerivation feature provides an additional facility to support automation of the primary management service in the process of filling in a list of parameter values from the underlying implementation.

One value of this facility is to provide an efficient means to get parameter values from the managed system. This support is implemented by a single call that returns all values for all parameters that can be derived from the model for a particular context.

Another value of this feature is that it provides a means for a generic management service to provide launch capability for auxiliary services that the management service is not specifically coded to handle.

The provisions that specify this optional feature are described in 7.1.4.

# 7 Implementation

This clause details the requirements and constraints related to the arrangement of instances and their properties for implementations of this profile. This clause documents only those classes and properties that have constraints or implementation requirements different from the corresponding element in either the CIM schema or a referenced profile that it modifies.

Unless otherwise specified, all properties of each supported class shall have values (possibly including null) as specified in the underlying schema. The reader is expected to be familiar with the provisions of that underlying schema.

Figure 5 shows the mandatory class adaptations of the Launch in Context profile.

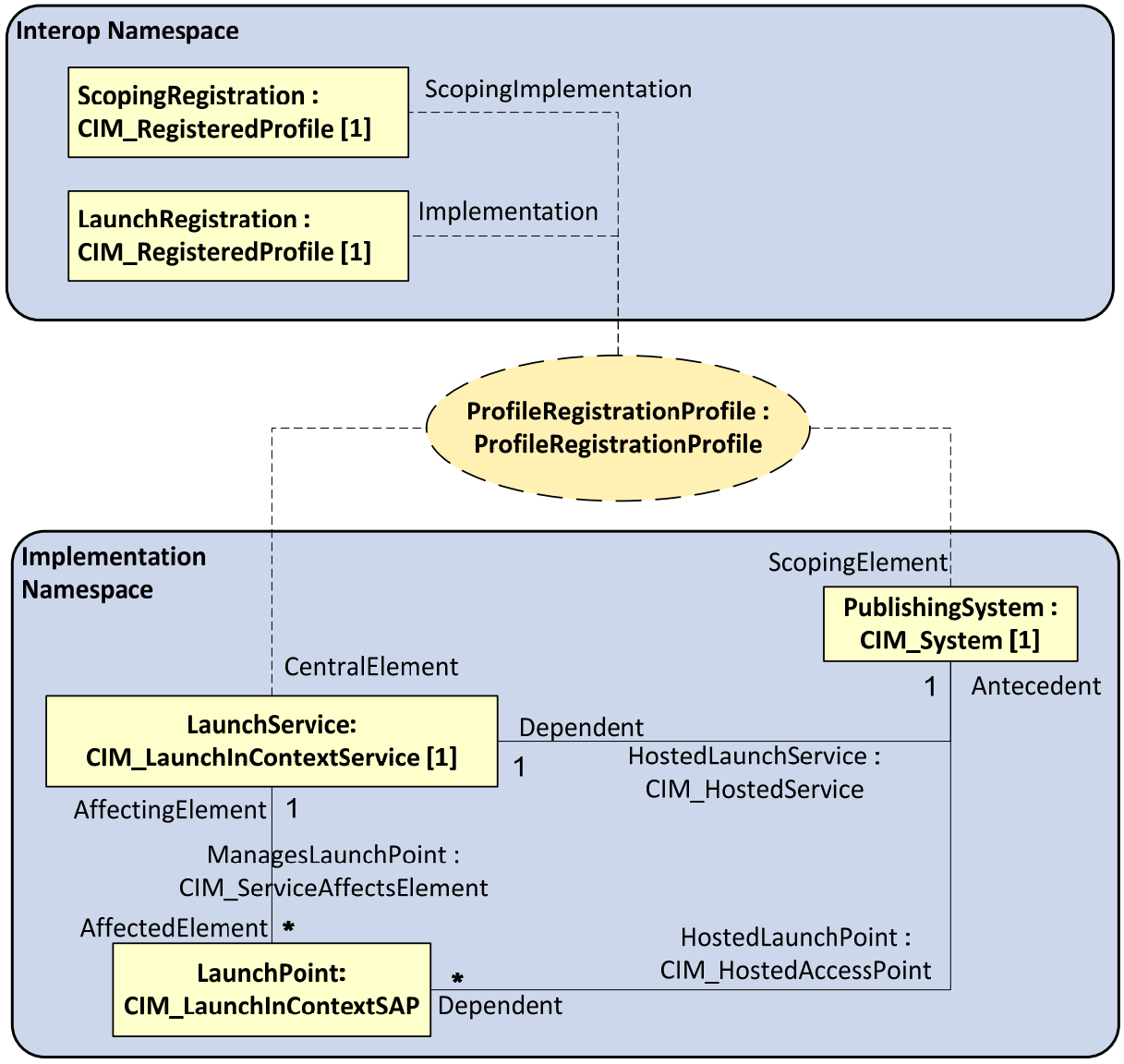



**Figure 5 – Mandatory adaptations of the Launch in Context profile**

## 7.1 Features

This clause specifies the features that extend the capabilities of this profile. Each of these features adds or extends to the set of mandatory adaptations as shown in Figure 5.

### 7.1.1 Feature: ScopedLaunchPoints

This clause defines the ScopedLaunchPoints feature.

#### 7.1.1.1 Feature description

Without this feature, each instance of CIM_LaunchInContextSAP applies to all instances that are a kind of one of the classes in the ManagedClasses array of that instance. Implementation of the ScopedLaunchPoints feature enables the scope of the CIM_LaunchInContextSAP to be further restricted to a specified set of other instances.

The implementation of the ScopedLaunchPoints feature is optional.

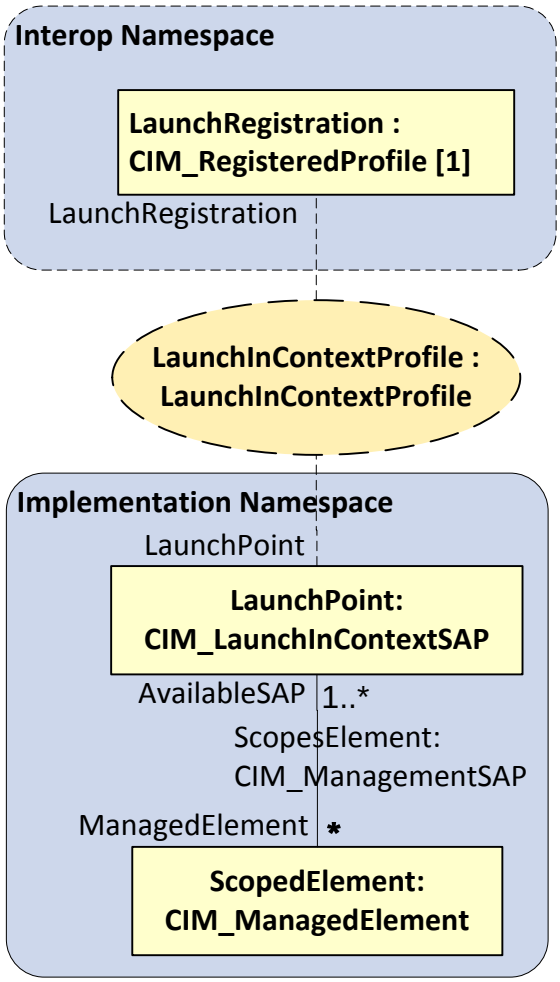



**Figure 6 – Modified and added adaptations for the ScopedLaunchPoints feature**

#### 7.1.1.2 Feature discovery

A management client can detect that the ScopedLaunchPoints feature is implemented by inspecting the CIM_RegisteredProfile instance representing this profile. This feature is implemented if one of the entries of the ImplementedFeatures array property has the value "ScopedLaunchPoints".

### 7.1.2 Feature: ClientPublishing

This clause defines the ClientPublishing feature.

#### 7.1.2.1 Feature description

The ClientPublishing feature is implemented to add, remove, or modify CIM_LaunchInContextSAP instances.

The implementation of the ClientPublishing feature is optional.

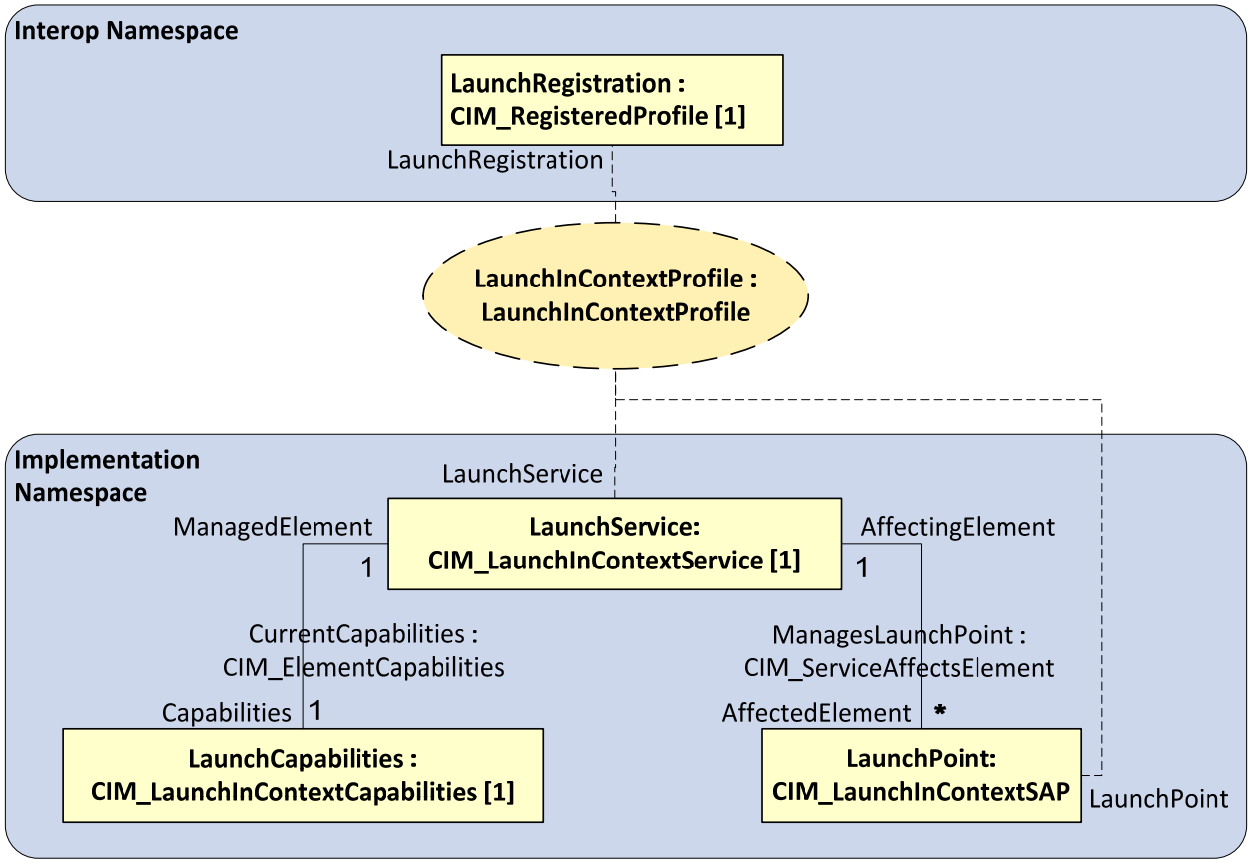



**Figure 7 – Added and modified adaptations for the ClientPublishing feature**

#### 7.1.2.2 Feature discovery

A management client can detect that the ClientPublishing feature is implemented by inspecting the CIM_RegisteredProfile instance representing this profile. This feature is implemented if one of the entries of the ImplementedFeatures array property has the value "ClientPublishing".

### 7.1.3 Feature: LaunchPointMonitoring

This clause defines the LaunchPointMonitoring feature.

#### 7.1.3.1 Feature description

Implementation of the LaunchPointMonitoring feature enables a management client to subscribe for creation, deletion, or modification indications for LaunchPoint instances.

The implementation of the LaunchPointMonitoring feature is optional.

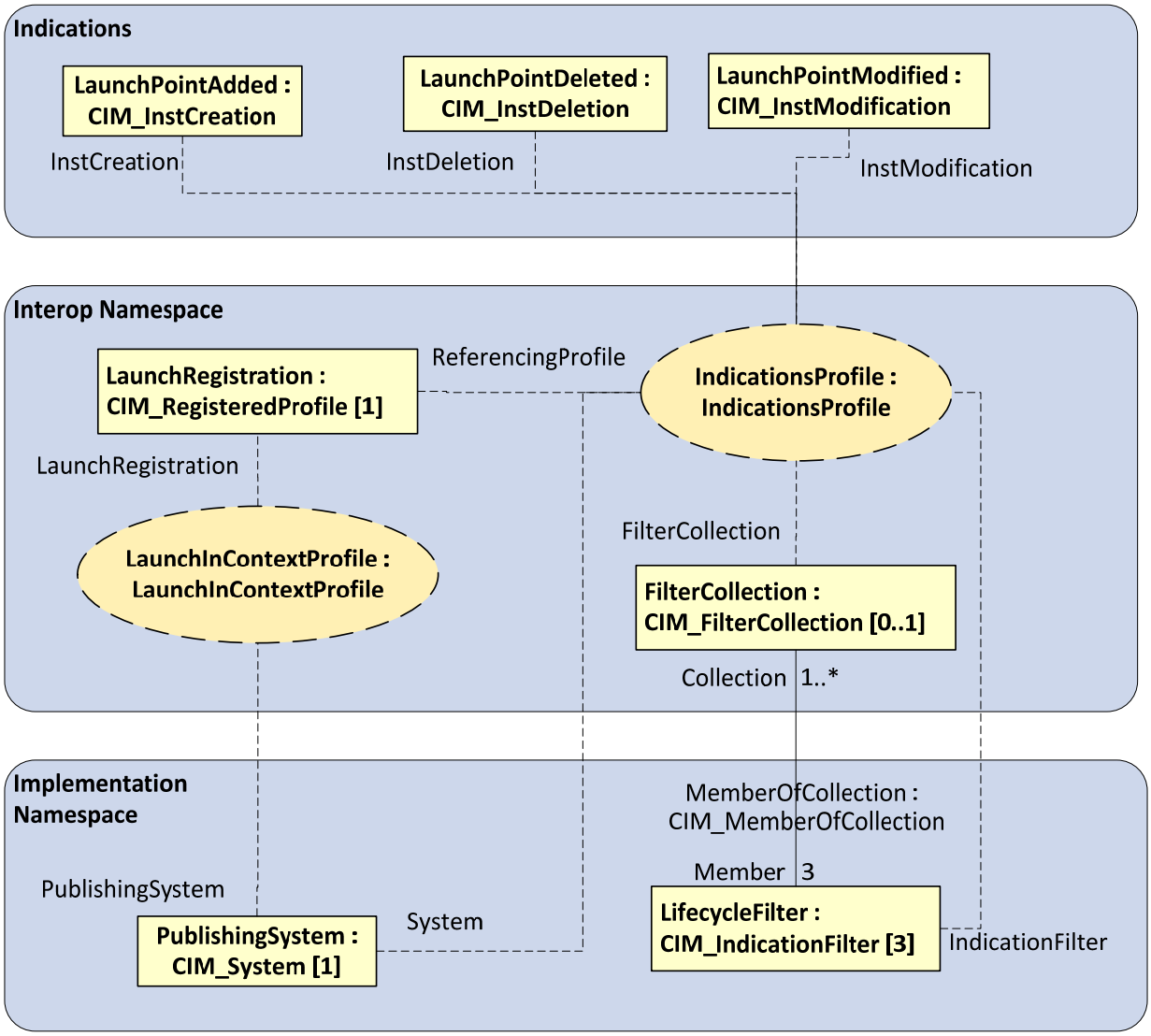



**Figure 8 – Added and modified adaptations for the LaunchPointMonitoring feature**

#### 7.1.3.2 Feature discovery

A management client can detect that the LaunchPointMonitoring feature is implemented by inspecting the CIM_RegisteredProfile instance representing this profile. This feature is implemented if one of the entries of the ImplementedFeatures array property has the value "LaunchPointMonitoring".

### 7.1.4 Feature: ParameterDerivation

This clause defines the ParameterDerivation feature.

#### 7.1.4.1 Feature description

Implementation of the ParameterDerivation feature provides assistance to the management client in producing parameter values for substitution into the URI template stored as the AccessInfo value of a CIM_LaunchInContextSAP instance.

The implementation of the ParameterDerivation feature is optional.

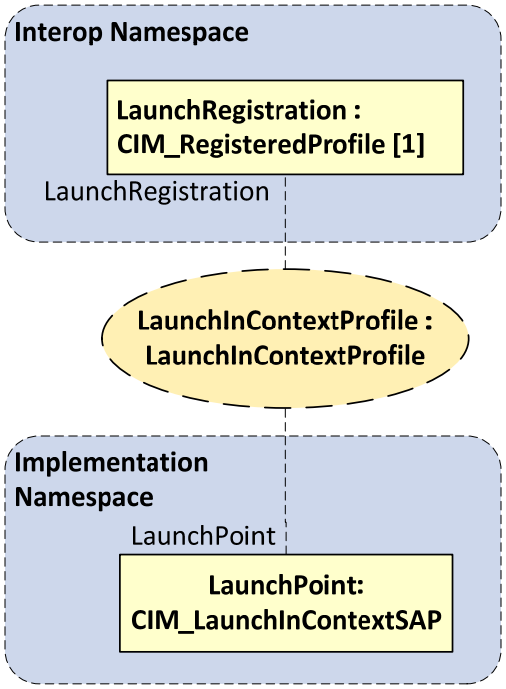



**Figure 9 – Modified adaptations for the ParameterDerivation feature**

#### 7.1.4.2 Feature discovery

A management client can detect that the ParameterDerivation feature is implemented by inspecting the CIM_RegisteredProfile instance representing this profile. This feature is implemented if one of the entries of the ImplementedFeatures array property has the value "ParameterDerivation".

## 7.2 Adaptations

### 7.2.1 General Requirements

Sets of instances perform various roles within this profile. Each set is named and is collectively referred to as a class adaptation (or simply an adaptation).

A class adaptation also defines implementation criteria including support criteria for various operations and methods of the class and constraints on the values of the properties and parameters of the class.

A reference to the adaptation name implies a reference to the class adaptation.

Table 4 defines implementation requirements for each of the class adaptations of this profile.

The reader is expected to be familiar with the provisions of the underlying schema (see CIM) and of implemented prerequisite profiles. The profile implementer is responsible for translating these requirements to the implementation infrastructure.

This profile defines operation requirements based on DSP0223.

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

Unless explicitly specified in the subclauses of 7.2, implementation requirements for methods for each adaptation (including those of associations) is as specified in the underlying schema (and as specified by implemented prerequisite profiles).

Unless explicitly specified in the subclauses of 7.2, implementation requirements for properties for each adaptation (including those of associations) is as specified in the underlying schema (and as specified by implemented prerequisite profiles).

### 7.2.2 Adaptation: CurrentCapabilities: CIM_ElementCapabilities

The CurrentCapabilities adaptation models the relationship between LaunchService (see 7.2.9) and LaunchCapabilities (see 7.2.7).

Implementation of the CurrentCapabilities adaptation is conditional.

Condition: The ClientPublishing (7.1.2) feature is implemented.

CurrentCapabilities instances shall conform to the requirements in Table 6 and the schema.

**Table 6 – CurrentCapabilities: CIM_ElementCapabilities**

| Elements | Requirement | Description |
|---|---|---|
| Characteristics [ ] | Mandatory | Value shall be 3 (Current). |
| Capabilities | Mandatory | Value shall reference an instance of LaunchCapabilities.<br>**Multiplicity: 0..1** |
| ManagedElement | Mandatory | Value shall reference an instance of LaunchService.<br>**Multiplicity: 1** |

The ClientPublishing feature requires a CurrentCapabilities instance for each LaunchService instance.

### 7.2.3 Adaptation: FilterCollection: CIM_FilterCollection

The FilterCollection adaptation models a collection of indication filters.

Implementation of the FilterCollection adaptation is conditional.

Condition: The LaunchPointMonitoring (7.1.3) feature is implemented.

Filter Collection instances shall conform to the requirements in Table 7 and to those defined in DSP1054 and the schema.

The LaunchPointMonitoring feature requires that if the FilterCollection is subscribed to and if a DMTF:LaunchPoint:Added, DMTF:LaunchPoint:Deleted, or DMTF:LaunchPoint:Modified event occurs, then a corresponding indication shall be delivered.

609 **Table 7 – FilterCollection: CIM_FilterCollection**

| Elements | Requirement | Description |
|---|---|---|
| CollectionName | See 7.1.3. | Value shall be "DMTF:LaunchPoint:Monitoring". |

610 The LaunchPointMonitoring feature requires exactly one instance of FilterCollection.

611 If the LaunchPointMonitoring feature is not implemented, there shall be no instances of FilterCollection.

### 612 7.2.4 Adaptation: FilterCollectionMember: CIM_MemberOfCollection

613 The FilterCollectionMember adaptation models the relationship between a FilterCollection instance and a
614 LifecycleFilter instance.

615 Implementation of the FilterCollectionMember adaptation is conditional.

616 Condition: The LaunchPointMonitoring (7.1.3) feature is implemented.

617 Each FilterCollectionMember shall have values that conform to the requirements in Table 8 and the
618 schema.

619 **Table 8 – FilterCollectionMember: CIM_MemberOfCollection**

| Elements | Requirement | Description |
|---|---|---|
| Collection | Mandatory | Shall reference the FilterCollection instance matching a row in Table 9. |
| Member | Mandatory | Shall reference the LifecycleFilter instance matching a row in Table 9.<br>**Multiplicity: 3** |

620 The LaunchPointMonitoring feature requires one instance of FilterCollectionMember for each instantiated
621 pair of FilterCollection and LifecycleFilter instances that match the same row of Table 9.

622 **Table 9 – FilterCollectionMember: CIM_MemberOfCollection instance pairs**

| LifecycleFilter.Name | FilterCollection.CollectionName |
|---|---|
| DMTF:LaunchPoint:Added | DMTF:LaunchPoint:Lifecycle |
| DMTF:LaunchPoint:Deleted | DMTF:LaunchPoint:Lifecycle |
| DMTF:LaunchPoint:Modified | DMTF:LaunchPoint:Lifecycle |

### 623 7.2.5 Adaptation: HostedLaunchPoint: CIM_HostedAccessPoint

624 The HostedLaunchPoint adaptation models the relationship between PublishingSystem (see 7.2.16) and
625 LaunchPoint (see 7.2.8).

626 Implementation of the HostedLaunchPoint adaptation is mandatory.

627 Each instance shall have values that conform to the requirements in Table 10 and the schema.

628 **Table 10 – HostedLaunchPoint: CIM_HostedAccessPoint element constraints**

| Elements | Requirement | Description |
|---|---|---|
| Antecedent | Mandatory | Value shall reference the PublishingSystem instance.<br>**Multiplicity: 1** |

| | | |
|---|---|---|
| Dependent | Mandatory | Value shall reference a LaunchPoint instance.<br>**Multiplicity: *** |

629 One HostedLaunchPoint instance shall be present for each LaunchPoint instance.

### 630 7.2.6 Adaptation: HostedService: CIM_HostedService

631 The HostedService adaptation models the relationship between LaunchService (see 7.2.9) and
632 PublishingSystem (see 7.2.16).

633 Implementation of the HostedService adaptation is mandatory.

634 Each instance shall have values that conform to the requirements in Table 11 and the schema.

635 **Table 11 – HostedService: CIM_HostedService element constraints**

| Elements | Requirement | Description |
|---|---|---|
| Antecedent | Mandatory | Shall be a reference to a PublishingSystem instance.<br>**Multiplicity: 1** |
| Dependent | Mandatory | Shall be a reference to a LaunchService instance.<br>**Multiplicity: 1** |

636 One HostedService instance shall be present for each LaunchService instance.

### 637 7.2.7 Adaptation: LaunchCapabilities: CIM_LaunchInContextCapabilities

638 The LaunchCapabilities adaptation models the capabilities of a LaunchService.

639 Implementation of the LaunchCapabilities adaptation is conditional.

640 Condition: The ClientPublishing (7.1.2) feature is implemented.

641 Each LaunchCapabilities instance shall conform to the requirements in Table 12 and the schema.

642 **Table 12 – LaunchCapabilities: CIM_LaunchInContextCapabilities**

| Elements | Requirement | Description |
|---|---|---|
| MaxRestrictionListSize | Mandatory | The value shall be set to a value that the implementation can support atomically in the context of a single invocation of one of those methods.<br>Zero indicates no defined limit. |
| MaxLaunchPoints | Mandatory | This value shall be the maximum number of LaunchPoint instances that may be instantiated at one time within the modeled implementation.<br>Zero indicates no defined limit. |

643 The ClientPublishing feature requires exactly one LaunchCapabilities instance for each LaunchService.

644 ### 7.2.8 Adaptation: LaunchPoint: CIM_LaunchInContextSAP

645 #### 7.2.8.1 General

646 The LaunchPoint adaptation models the information published by the PublishingSystem that provides the
647 information necessary for the administrator or management service to launch auxiliary services. Those
648 auxiliary services provide additional management features for a set of resources known on the publishing
649 system.

650 NOTE 1: While those resources are known in the context of the PublishingSystem, they may be part of some other
651 system.

652 NOTE 2: CIM_LaunchInContextSAP is subclassed from CIM_RemoteServiceAccessPoint because the auxiliary
653 management service that it addresses is assumed to be hosted by some other system.

654 NOTE 3: The named auxiliary management service may be hosted on the publishing system.
655 A client should use values of the SupportedFeatureName and SupportedFeatureDescription properties (see 7.2.8.8
656 and 7.2.8.9) for selecting a particular LaunchPoint. Use of the Name property as defined in the schema is not
657 recommended for this purpose.

658 Implementation of the LaunchPoint adaptation is mandatory.

659 Each LaunchPoint instance shall conform to the requirements in Table 13 and the schema.

660 Each LaunchPoint shall be associated by a HostedLaunchPoint instance to its PublishingSystem.

661 **Table 13 – LaunchPoint: CIM_LaunchInContextSAP**

| Elements | Requirement | Description |
|---|---|---|
| AccessContext | Mandatory | The value shall be set to 10 (Management Service). |
| AccessInfo | Mandatory | Usage: See 7.2.8.2. |
| InfoFormat | Mandatory | The value be set to either 200 (URL) or 206 (ParameterizedURL). |
| ParameterName [ ] | Mandatory | Usage: See 7.2.8.3. |
| ParameterDescription [ ] | Optional | Usage: See 7.2.8.4. |
| ParameterConstraints [ ] | Optional | Usage: See 7.2.8.5. |
| ParameterType [ ] | Mandatory | Usage: See 7.2.8.6. |
| ParameterDerivation [ ] | Conditional | Condition: ParameterDerivation (see 7.1.4)<br>Usage: See 7.2.8.7. |
| SupportedFeatureName [ ] | Mandatory | Usage: See 7.2.8.8. |
| SupportedFeatureDescription [ ] | Mandatory | Usage: See 7.2.8.9. |
| ManagedClasses [ ] | Optional | Usage: See 7.2.8.10. |
| ManagementIsRestricted | Conditional | Condition: ScopedLaunchPoints (see 7.1.1)<br>Usage: See 7.2.8.11. |
| LaunchMessage | Optional | Usage: See 7.2.8.12. |
| LaunchMessageProtocolOperation | Mandatory | Usage: See 7.2.8.13. |
| GetDerivedParametersFor Element( ) | Conditional | Condition: ParameterDerivation (see 7.1.4)<br>Usage: See 7.2.8.14. |
| GetAssociatedInstancesWithPath( ) | Conditional | Condition: ClientPublishing (see 7.1.2)<br>Usage: See 7.2.8.15. |

| Elements | Requirement | Description |
|---|---|---|
| GetAssociatedInstancePaths( ) | Conditional | Condition: ScopedLaunchPoints (see 7.1.1)<br>Usage: See 7.2.8.16. |
| GetReferencingInstancesWith Path( ) | Conditional | Condition: ClientPublishing (see 7.1.2)<br>Usage: See 7.2.8.17. |

Instances of the LaunchPoint adaptation are optional.

#### 7.2.8.2 Property: AccessInfo

The value of the AccessInfo property shall contain a URI that may contain ${parameterName} references.

The primary management service is responsible for replacing each parameterized reference with a value to create a URI that may then be utilized in an HTTP GET request if used alone, or in a protocol operation defined by LaunchRequestProtocolOperation if used in conjunction with the Message Based Launch feature.

The primary management service is responsible for replacing each *parameterName* entry with a value to create a URI that may then be utilized in requests defined by LaunchMessageProtocolOperation.

How a primary management service obtains the replacement values depends on the entries in the ParameterName and related arrays. Each *parameterName* entry in the template URI shall have a matching entry in the ParameterName array.

Without benefit of the ParameterDerivation feature (see 7.1.4) and based on its understanding of the named parameter, the primary management service shall obtain the values programmatically or from its clients for each *parameterName* entry. The primary management service is expected to enforce type and other constraints expressed in the ParameterType (see 7.2.8.6) and ParameterConstraint (see 7.2.8.5) properties.

The ParameterDerivation feature recommends that the primary management service use the following algorithm to determine the values of each *parameterName* in the URI:

1) Allocate an empty ParameterValues array, with entries corresponding to the entries of the ParameterName array.
2) If any entry in the ParameterDerivation array is not NULL, select a resource (represented by an instance of CIM_ManagedElement) that acts as the focal point for deriving context.
3) Invoke the GetDerivedParametersForElement( ) method to update the ParameterValues array.
4) For each empty value in the ParameterDerivation array, the primary management client should utilize the corresponding values from the ParameterName, ParameterDescription, ParameterType, and ParameterConstraint properties as input to obtain the corresponding ParameterValues entry programmatically or from its clients. The primary management service is expected to enforce type and other constraints expressed in the ParameterType (see 7.2.8.6) and ParameterConstraint (see 7.2.8.5).
5) The values for each *parameterName* are retrieved from the resultant ParameterValues array entry.

#### 7.2.8.3 Property: ParameterName

ParameterName is an array with zero or more entries. The value of each entry specifies the name of a parameter specified in the URL string stored in the AccessInfo property as "${*parameterName*}".

The value of *parameterName* should have the format *orgName*":"*specName*":"*specVersion*":"*localName*. In this format, *orgName* shall be a trademarked or otherwise owned name of the defining organization,

*specName* together with *specVersion* shall name a specification within that organization, and *localName* shall be a name defined by the specification.

For DMTF-defined parameters, *orgName* shall be "DMTF", *specName* shall be the DSP name of the specification defining the parameter, and *specVersion* shall have the form *m*"."*n* where *m* is the major version number and *n* is the minor version number. Each number shall not include leading zeros.

#### 7.2.8.4 Property: ParameterDescription

ParameterDescription is an array, with entries correlated to the entries of ParameterName. The value of each entry should provide information about the corresponding parameter, which may be displayed in a user interface.

#### 7.2.8.5 Property: ParameterConstraints

ParameterConstraints is an array, with entries that correlate to the entries of ParameterName. If the parameter value can be any value in the range defined by the ParameterType, the value of the corresponding PropertyConstraint entry is empty. Otherwise, the value of the PropertyConstraint entry should contain an OCL Invariant constraint that limits the values of the string. (See A.3 for details.)

#### 7.2.8.6 Property: ParameterType

ParameterType is an array, with entries correlated to the entries of ParameterName. The value of each entry specifies the type of the corresponding entry in the ParameterName property. If ParameterType is not specified, 3 (string) is assumed.

#### 7.2.8.7 Property: ParameterDerivation

Implementation of the ParameterDerivation property is conditional.

Condition: The ParameterDerivation feature (see 7.1.4) is implemented.

If the ParameterDerivation feature is not implemented, this property shall be null or the value of each entry shall be empty.

The ParameterDerivation feature (see 7.1.4) requires that the value of each entry of ParameterDerivation shall be empty or shall specify an OCL 2.0 derivation string for the corresponding ParameterName array entry value (see Annex A).

The GetDerivedParametersForElement( ) method evaluates each derivation expression; for details of expression evaluation and more information on this method, see 7.2.8.14.

#### 7.2.8.8 Property: SupportedFeatureName

SupportedFeatureName is an array of zero or more entries, the value of each entry shall name a feature supported by the service addressed by AccessInfo (see 7.2.8.2).

At least one non-empty entry shall be present. Feature names shall have the format *orgName*":"*specName*":"*specVersion*":"*featureName*, where *orgName* shall be a trademarked or otherwise owned name of the defining organization, *specName* together with *specVersion* shall name a specification within that organization, and featureName shall be a name defined by the specification.

For DMTF-defined features, *orgName* shall be "DMTF"; *specName* shall be the DSP name of a management profile, and *specVersion* shall have the form *m*"."*n* where *m* is the major version number and *n* is the minor version number. Each number shall not include leading zeros.

#### 7.2.8.9 Property: SupportedFeatureDescription

The value of each non-empty entry should provide information useful to an end user as an aid to understanding the corresponding feature. Any use of parameters should be explained in the description of the feature. A non-empty entry in SupportedFeatureDescription shall be present for each corresponding non-empty entry of SupportedFeatureName.

#### 7.2.8.10 Property: ManagedClasses

This property is used to define a set of instances. If ManagedClasses is NULL, all instances belong to the set. If ManagedClasses is not NULL, then the value of each entry names a class and all instances of that class are included in the set. The set of instances defines the range of instances that the auxiliary management service addressed by this LaunchPoint may manage, subject to the following restrictions:

- If the ScopedLaunchPoints feature (see 7.1.1) is not implemented or if ManagementIsRestricted is False, all instances in the set that also belong to the same CIM Namespace as the LaunchPoint may be managed.
- The ScopedLaunchPoints feature (see 7.1.1) is implemented and if ManagementIsRestricted value is True, then only instances in the set that are also associated to this instance by the ManagesLaunchPoint association may be managed. In this case, associated instances of classes belonging to the named set of classes may belong to any CIM Namespace.

#### 7.2.8.11 Property: ManagementIsRestricted

Implementation of the ManagementIsRestricted property is conditional.

Condition: The ScopedLaunchPoints feature is implemented.

If the ScopedLaunchPoints feature (see 7.1.1) is not implemented, the value of this property shall be set to False.

If the ScopedLaunchPoints feature (see 7.1.1) is implemented, the value of this property may be True. If True, it indicates that only those instances associated to this instance by the ManagesLaunchPoint association may be managed by the auxiliary management service named in the AccessInfo property.

#### 7.2.8.12 Property: LaunchMessage

The value of LaunchMessage is a template for a message (that typically represents a request) to be sent to the URI specified in the AccessContext property.

LaunchMessage shall be NULL if the associated operation specified by LaunchMessageProtocolOperation (see 7.2.8.13) does not send a message.

Like the AccessInfo property, this template may contain placeholders as ${*ParameterName*}, where *ParameterName* matches an entry in the ParameterName property. The format of the template is defined by the protocol specified by the value of the LaunchMessageProtocolOperation (see 7.2.8.13).

See the description in AccessInfo (7.2.8.2) for the primary management service algorithm required to get values to replace the placeholders.

#### 7.2.8.13 Property: LaunchMessageProtocolOperation

The value of LaunchMessageProtocolOperation is an enumeration that defines the protocol operation that the primary management service is expected to use when making a request formed from the value in the LaunchMessage property.

NOTE: No provisions of this profile support processing or interpretation of responses from the launched service. This profile is not intended to create a general facility for invoking methods.

778 #### 7.2.8.14 Method: GetDerivedParametersForElement( )

779 Implementation of the GetDerivedParameterForElement( ) method is conditional.

780 Condition: The ParameterDerivation feature is implemented.

781 The ParameterDerivation feature (see 7.1.4) requires the implementation to support the input and output
782 parameters and return values specified by the schema and as additionally constrained by this clause.

783 When executed, the implementation shall compute values for those parameters that have a derivation
784 specified in the corresponding entry of the ParameterDerivation property (see Annex A).

785 **Table 14 – LaunchPoint: GetDerivedParametersForElement( ): Parameters**

| Parameter Name | Description |
|---|---|
| **Input Parameters** | |
| Self | A reference to the managed element referred to as "self" in derivation expressions contained in the ParameterDerivation array |
| **Output Parameters** | |
| ParameterValue [ ] | An array of parameter values corresponding to the entries of the ParameterNames array |
| ReturnValue | Expected value is 0 (see CIM Schema). |

786 #### 7.2.8.15 Operation: GetAssociatedInstancesWithPath( )

787 Implementation of the GetAssociatedInstancesWithPath( ) operation is conditional.

788 Condition: The ClientPublishing feature is implemented.

789 The ClientPublishing feature (see 7.1.2) requires the GetAssociatedInstancesWithPath( ) operation to
790 support the input and output parameters and messages as specified in Table 15 and in DSP0223.

791 The purpose of this usage is to enable the primary management service to get the LaunchService that
792 manages a LaunchPoint.

793 **Table 15 – LaunchPoint: GetAssociatedInstancesWithPath( ): Parameters**

| Parameter Name | Description |
|---|---|
| **Input Parameters** | |
| SourceInstancePath | For this use, the value shall be the instance path of a LaunchPoint adaptation. |
| AssociationClassName | For this use, the value shall be "CIM_ServiceAffectsElement". |
| AssociatedClassName | For this use, the value shall be "CIM_LaunchInContextService". |
| **Output Parameters** | |
| InstanceList [ ] | For this use, the value shall contain exactly one LaunchService adaptation instance. |

794 #### 7.2.8.16 Operation: GetAssociatedInstancePaths( )

795 Implementation of the GetAssociatedInstancePaths( ) operation is conditional.

796 Condition: The ScopedLaunchPoints feature is implemented.

797 The ScopedLaunchPoints feature (see 7.1.1) requires the GetAssociatedInstancePaths( ) operation to
798 support the input and output parameters and messages specified in this clause and in DSP0223.

799 The purpose of this usage is to enable listing Elements scoped to a LaunchPoint.

800 **Table 16 – LaunchPoint: GetAssociatedInstancePaths( ): Parameters**

| Parameter Name | Description |
|---|---|
| **Input Parameters** | |
| SourceInstancePath | For this use, the value shall be the instance path of a LaunchPoint adaptation. |
| AssociationClassName | For this use, the value shall be "CIM_ManagementSAP". |
| **Output Parameters** | |
| InstancePathList [ ] | For this use, the value shall contain zero or more ScopedElement adaptation instances. |

801 #### 7.2.8.17 Operation: GetReferencingInstancesWithPath( )

802 Implementation of the GetReferencingInstancesWithPath( ) operation is conditional.

803 Condition: The ClientPublishing feature is implemented.

804 The ClientPublishing feature (see 7.1.2) requires the GetReferencingInstancesWithPath( ) operation to
805 support the input and output parameters and messages specified in this clause and in DSP0223.

806 The purpose of this usage is to enable the primary management service to get the ManagesLaunchPoint
807 association instances that connect a LaunchPoint to its managing LaunchService instances.

808 NOTE: The caller is expected to match up instance paths of CIM_LaunchInContextService instances returned above
809 with the instance path returned in the CIM_ServiceAffectsElement.AffectingElement property returned from this
810 action.

811 **Table 17 – LaunchPoint: GetReferencingInstancesWithPath( ): Parameters**

| Parameter Name | Description |
|---|---|
| **Input Parameters** | |
| SourceInstancePath | For this use, the value shall be the instance path of a LaunchPoint adaptation. |
| AssociationClassName | For this use, the value shall be "CIM_ServiceAffectsElement". |
| AssociatedClassName | For this use, the value shall be "CIM_LaunchInContextService". |
| **Output Parameters** | |
| InstanceList [ ] | For this use, the value shall contain exactly one ManagesLaunchPoint adaptation instance. |

812 ### 7.2.9 Adaptation: LaunchPointAdded: CIM_InstCreation

813 The LaunchPointAdded adaptation specifies the information transmitted in the indication of a
814 LaunchPointAdded event.

815 Implementation of the LaunchPointAdded adaptation is conditional.

816 Condition: The LaunchPointMonitoring (7.1.3) feature is implemented.

817 Each LaunchPointAdded instance shall conform to the requirements in Table 18, the Indications profile
818 (DSP1054), and the schema.

819 **Table 18 – LaunchPointAdded: CIM_InstCreation**

| Elements | Requirement | Description |
|---|---|---|
| IndicationFilterName | Mandatory | "DMTF:LaunchPoint:Added" |
| PerceivedSeverity | Mandatory | The value shall be 2 (Information). |

820 The LaunchPointMonitoring feature requires that if LaunchPoint Added events are subscribed for, and if a
821 LaunchPoint Added event occurs, then LaunchPointAdded indications shall be generated.

822 ### 7.2.10 Adaptation: LaunchPointDeleted: CIM_InstDeletion

823 The LaunchPointDeleted adaptation specifies the information transmitted in the indication of a
824 LaunchPointDeleted event.

825 Implementation of the LaunchPointDeleted adaptation is conditional.

826 Condition: The LaunchPointMonitoring (7.1.3) feature is implemented.

827 Each LaunchPointDeleted instance shall conform to the requirements in Table 19, the Indications profile
828 (DSP1054), and the schema.

829 **Table 19 – LaunchPointDeleted: CIM_InstDeletion**

| Elements | Requirement | Description |
|---|---|---|
| IndicationFilterName | Mandatory | "DMTF:LaunchPoint:Deleted" |
| PerceivedSeverity | Mandatory | The value shall be 2 (Information). |

830 The LaunchPointMonitoring feature requires that if LaunchPoint Deleted events are subscribed for, and if
831 a LaunchPoint Deleted event occurs, then LaunchPointDeleted indications shall be generated.

832 ### 7.2.11 Adaptation: LaunchPointModified: CIM_InstModification

833 The LaunchPointModified adaptation specifies the information transmitted in the indication of a
834 LaunchPointModified event.

835 Implementation of the LaunchPointModified adaptation is conditional.

836 Condition: The LaunchPointMonitoring (7.1.3) feature is implemented.

837 Each LaunchPointModified instance shall conform to the requirements in Table 20, the Indications profile
838 (DSP1054), and the schema.

839 **Table 20 – LaunchPointModified: CIM_InstModification**

| Elements | Requirement | Description |
|---|---|---|
| IndicationFilterName | Mandatory | "DMTF:LaunchPoint:Modified" |
| PerceivedSeverity | Mandatory | The value shall be 2 (Information). |

840 The LaunchPointMonitoring feature requires that if LaunchPoint Modified events are subscribed for, and if
841 a LaunchPoint Modified event occurs, then LaunchPointModified indications shall be generated.

### 7.2.12 Adaptation: LaunchProfile: CIM_RegisteredProfile

#### 7.2.12.1 General

The LaunchProfile adaptation is used to publish conformance to this profile.

Implementation of the LaunchProfile adaptation is mandatory.

Each LaunchProfile instance shall conform to the requirements in Table 21, the Profile Registration profile (DSP1033), and the schema.

The LaunchProfile instance shall be associated directly or indirectly to each LaunchService instance, to each LaunchSystem instance, and to at least one ScopingProfile instance by a means specified by the Profile Registration profile (DSP1033).

**Table 21 – LaunchProfile: CIM_RegisteredProfile**

| Elements | Requirement | Description |
|---|---|---|
| RegisteredName | Mandatory | The RegisteredName property shall contain "LaunchInContext". |
| RegisteredOrganization | Mandatory | The RegisteredOrganization property shall contain "DMTF". |
| RegisteredVersion | Mandatory | The RegisteredVersion property shall be the major and minor version of this profile and shall have the form *m*"."*n* where *m* is the major version number and *n* is the minor version number. Each number shall not include leading zeros. |
| ImplementedFeatures | Conditional | Condition: ScopedLaunchPoints (see 7.1.1)<br>One entry shall contain a value equal to "ScopedLaunchPoints". |
| | Conditional | Condition: ClientPublishing (see 7.1.2)<br>One entry shall contain a value equal to "ClientPublishing". |
| | Conditional | Condition: LaunchPointMonitoring (see 7.1.3)<br>One entry shall contain a value equal to "LaunchPointMonitoring". |
| | Conditional | Condition: ParameterDerivation (see 7.1.4)<br>One entry shall contain a value equal to "ParameterDerivation". |
| OpenConformantInstances() | Optional | See 7.2.12.3. |
| CloseConformantInstances() | Optional | See 7.2.12.2 where enumerationContext value is as defined in 7.2.12.3. |
| PullConformantInstances() | Optional | See 7.2.12.4 where the instances are LaunchService adaptations and the enumerationContext value is as defined in 7.2.12.3 or by subsequent invocations for that enumerationContext. |

A system that wishes to advertise conformance to this profile shall instantiate an instance of LaunchProfile in the Interop namespace.

#### 7.2.12.2 Method: CloseConformantInstances( )

The implementation of the CloseConformantInstances( ) method is optional. An implementation should support the CloseConformantInstances( ) method as specified by this clause and by the schema.

The purpose of this usage is to enable the CIM client to close a session created by OpenConformantInstances( ).

#### 7.2.12.3 Method: OpenConformantInstances( )

The implementation of the OpenConformantInstances( ) method is optional. An implementation should support the OpenConformantInstances( ) method as specified by this clause and by the schema.

The purpose of this usage is to start listing the central (LaunchService) instances that are conformant to this profile.

Parameters for this method are listed in Table 22

**Table 22 – LaunchProfile: OpenConformantInstances( ): Parameters**

| Parameter Name | Description |
|---|---|
| **Input Parameters** | |
| ResultClass | For this use, the value shall be "CIM_LaunchInContextService". |
| **Output Parameters** | |
| InstanceWithPathList [ ] | For this use, the value shall contain zero or more LaunchService instances. |

#### 7.2.12.4 Method: PullConformantInstances( )

The implementation of the PullConformantInstances( ) method is optional. An implementation should support the PullConformantInstances( ) method as specified by this clause and by the schema.

The purpose of this usage is to provide the ability to continue listing the central (LaunchService) instances that are conformant to this profile.

Parameters for this method are listed in Table 23.

**Table 23 – LaunchProfile: PullConformantInstances( ): Parameters**

| Parameter Name | Description |
|---|---|
| **Input Parameters** | |
| EnumerationContext | For this use, the value shall be a value returned from the most recent execution of 7.2.12.3 or 7.2.12.4. |
| **Output Parameters** | |
| InstanceWithPathList [ ] | For this use, the value shall contain zero or more LaunchService instances. |

### 7.2.13 Adaptation: LaunchService: CIM_LaunchInContextService

#### 7.2.13.1 General

Implementation of the LaunchService adaptation is mandatory.

The LaunchService adaptation models the ability of the PublishingSystem to publish LaunchPoints.

Each LaunchService instance shall be conformant to the requirements in Table 24 and to the schema.

878 **Table 24 – LaunchService: CIM_LaunchInContext**

| Elements | Requirement | Description |
|---|---|---|
| CreateLaunchPoint( ) | Condition | Condition: ClientPublishing (see 7.1.2)<br>See 7.2.13.2. |
| RemoveLaunchPoint( ) | Condition | Condition: ClientPublishing (see 7.1.2)<br>See 7.2.13.3. |
| AssertLaunchPoint( ) | Condition | Condition: ClientPublishing (see 7.1.2)<br>See 7.2.13.4. |
| GetAssociatedInstancesWith Path() | Condition | Condition: ClientPublishing (see 7.1.2)<br>See 7.2.13.5. |

879 There shall be at least one LaunchService instance.

880 **7.2.13.2 Method: CreateLaunchPoint( )**

881 Implementation of the CreateLaunchPoint( ) method is conditional.

882 Condition: The ClientPublishing (see 7.1.2) feature is implemented.

883 The purpose of this usage is to enable the management client to create an instance of the LaunchPoint
884 adaptation and related associations.

885 **7.2.13.3 Method: RemoveLaunchPoint( )**

886 Implementation of the RemoveLaunchPoint( ) method is conditional.

887 Condition: The ClientPublishing (see 7.1.2) feature is implemented.

888 The purpose of this usage is to enable a management client to remove a LaunchPoint or to remove
889 ScopedElements from its scope.

890 If the RestrictedToElement array is NULL, then the removal of the referenced LaunchPoint is
891 unconditional and additionally removes all related ScopedElement and ManagesLaunchPoint
892 associations. Otherwise, the ScopedElement associations are removed and the LaunchPoint instance
893 and related ManagesLaunchPoint association are removed only if no ScopedElement associations
894 remain.

895 **7.2.13.4 Method: AssertLaunchPoint( )**

896 Implementation of the AssertLaunchPoint( ) method is conditional.

897 Condition: The ClientPublishing (see 7.1.2) feature is implemented.

898 The purpose of this usage is to enable a management client add additional ScopedElement instances to
899 the scope of a LaunchPoint instance.

900 **7.2.13.5 Operation: GetAssociatedInstancesWithPath( )**

901 Implementation of the GetAssociatedInstancesWithPath( ) operation is conditional.

902 Condition: The ClientPublishing (see 7.1.2) feature is implemented.

903 This operation provides the ability to list either the LaunchCapabilities or the LaunchPoints associated
904 with a LaunchService.

905 The parameters for this operation are listed in Table 25.

906 **Table 25 – LaunchService: GetAssociatedInstancesWithPath( ): Parameters**

| Name | Description |
|---|---|
| **Input Parameters** | |
| SourceInstancePath | The value shall be the instance path of a LaunchService adaptation. |
| AssociationClassName | To list the LaunchCapabilities, the value shall be "CIM_ElementCapabilities". |
| | To list LaunchPoints, the value shall be "CIM_ServiceAffectsElement". |
| AssociatedClassName | To list the LaunchCapabilities, the value shall be "CIM_LaunchInContextCapabilities". |
| | To list LaunchPoints, the value shall be "CIM_LaunchInContextSAP". |
| **Output Parameters** | |
| InstanceList [ ] | To list the LaunchCapabilities, the value shall contain zero or one LaunchCapabilities adaptation instances. |
| | To list LaunchPoints, the value shall contain zero or more LaunchPoint adaptation instances. |

907 ### 7.2.14 Adaptation: LifecycleFilter: CIM_IndicationFilter

908 Implementation of the LifecycleFilter adaptation is conditional.

909 Condition: The LaunchPointMonitoring (7.1.3) feature is implemented.

910 The LaunchPointMonitoring feature requires the LifecycleFilter to conform to the requirements in Table 26
911 and the schema.

912 **Table 26 – LifecycleFilter: CIM_IndicationFilter**

| Elements | Requirement | Description |
|---|---|---|
| Name | Mandatory | The value shall be defined by the LifecycleFilter.Name column of a row in Table 27. |
| Query | Mandatory | The value shall be defined by the LifecycleFilter.Query column of a row in Table 27. |
| QueryLanguage | Mandatory | The value shall be "DMTF:CQL". |

913 The LaunchPointMonitoring feature requires exactly three LifecycleFilter instances, one for each event
914 defined in Table 3. Each instance corresponds to one row of Table 27.

915 **Table 27 – LifecycleFilter: CIM_IndicationFilter property value constraints**

| LifecycleFilter.Name | LifecycleFilter.Query |
|---|---|
| The value shall be "DMTF:LaunchPoint:Added". | The value shall be "SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_LaunchInContextSAP". |
| The value shall be "DMTF:LaunchPoint:Deleted". | The value shall be "SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_LaunchInContextSAP". |
| The value shall be "DMTF:LaunchPoint:Modified". | The value shall be "SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LaunchInContextSAP". |

916 ### 7.2.15 Adaptation: ManagesLaunchPoint: CIM_ServiceAffectsElement

917 Implementation of the ManagesLaunchPoint adaptation is Mandatory.

The ManagesLaunchPoint adaptation models the relationship between a LaunchService and a LaunchPoint that it manages.

Each ManagesLaunchPoint instance shall have values that conform to the requirements in Table 28 and the schema.

**Table 28 – ManagesLaunchPoint: CIM_ServiceAffectsElement**

| Elements | Requirement | Description |
|---|---|---|
| AffectedElement | Mandatory | The value shall reference a LaunchPoint instance. |
| AffectingElement | Mandatory | The value shall reference the LaunchService instance. |
| ElementEffects | Mandatory | One element of the property ElementEffects shall contain the value 5 (Manages). |

There shall be one instance of ManagesLaunchPoint for each LaunchPoint instance.

## 7.2.16 Adaptation: PublishingSystem: CIM_System

### 7.2.16.1 General

Implementation of the PublishingSystem adaptation is Mandatory.

The PublishingSystem models a primary managed system that is capable of publishing LaunchPoints. Each PublishingSystem typically represents a top level system like a host, a storage array, a fibre channel switch, or a fabric.

The PublishingSystem instance shall be the central instance of the autonomous profile named by the ScopingProfile. The implementation shall publish conformance with the ScopingProfile and the LaunchProfile as specified by the Profile Registration profile (DSP1033).

The PublishingSystem instance shall conform to the requirements in Table 29 and the schema.

**Table 29 – PublishingSystem: CIM_System element constraints**

| Elements | Requirement | Description |
|---|---|---|
| GetAssociatedInstancesWithPath( ) | Mandatory | See 7.2.16.2. |

There shall be a PublishingSystem instance.

### 7.2.16.2 Operation: GetAssociatedInstancesWithPath( )

Implementation of the GetAssociatedInstancesWithPath( ) operation is mandatory.

The implementation shall support the GetAssociatedInstancesWithPath( ) operation as specified by this clause and by DSP0223.

The purpose of this usage is to enable the primary management service to follow the conformance path from the PublishingSystem to instances of the LaunchPoint adaptation or to instances of the LaunchService adaptation.

The parameters for this operation are listed in Table 30.

944 **Table 30 – PublishingSystem: GetAssociatedInstancesWithPath( ): Parameters**

| Parameter Name | Description |
|---|---|
| **Input Parameters** | |
| AssociationClassName | To list LaunchServices: The value shall contain one or more LaunchService instances. |
| | To list LaunchServices: The value shall be "CIM_HostedService". |
| AssociatedClassName | To list published LaunchPoints: The value shall be "CIM_LaunchInContextSAP". |
| | To list LaunchServices: The value shall be "CIM_LaunchInContextService". |
| **Output Parameters** | |
| InstanceList [ ] | To list published LaunchPoints: The value shall contain zero or more LaunchPoint instances. |

945 ### 7.2.17 Adaptation: ScopedElement: CIM_ManagedElement

946 #### 7.2.17.1 General

947 Implementation of the ScopedElement adaptation is conditional.

948 Condition: The ScopedLaunchPoints (7.1.1) feature is implemented.

949 The ScopedElement adaptation models a resource that may be managed by an auxiliary management
950 service that is advertised by a LaunchPoint. Each ScopedElement is connected to one or more
951 LaunchPoints by ScopesElement associations. For each such connected LaunchPoint instance, both of
952 the following requirements are in effect:

953 - The ManagementIsRestricted property shall be True.
954 - The ScopedElement shall be a kind of at least one of the classes named in the
955 ManagedClasses property.

956 Each ScopedElement instance shall be conformant to the requirements in Table 31 and the schema.

957 **Table 31 – ScopedElement: CIM_ManagedElement**

| Elements | Requirement | Description |
|---|---|---|
| GetAssociatedInstancesWithPath( ) | Conditional | Condition: Implementation of the ScopedLaunchPoints feature (see 7.1.1)<br>See 7.2.17.2. |

958 ScopedElement instances shall not be instantiated if the ScopedLaunchPoints feature is not implemented
959 (see 7.1.1).

960 #### 7.2.17.2 Operation: GetAssociatedInstancesWithPath( )

961 Implementation of the GetAssociatedInstancesWithPath( ) operation is conditional.

962 Condition: The ScopedLaunchPoints (see 7.1.1) feature is implemented.

963 If the ScopedLaunchPoints feature is implemented, the GetAssociatedInstancesWithPath( ) operation
964 shall support the input and output parameters and messages specified in this clause and in DSP0223.

965 The purpose of this usage is to provide the ability to list the LaunchPoints of a ScopedElement.

966 The parameters for this operation are listed in Table 32.

967 **Table 32 – ScopedElement: GetAssociatedInstancesWithPath( ): Parameters**

| Parameter Name | Description |
|---|---|
| **Input Parameters** | |
| SourceInstancePath | For this use, the value shall be the instance path of a ScopedElement adaptation. |
| AssociationClassName | For this use, the value shall be "CIM_ManagementSAP". |
| AssociatedClassName | For this use, the value shall be "CIM_LaunchInContextSAP". |
| **Output Parameters** | |
| InstanceList [ ] | For this use, the value shall contain zero or more LaunchPoint adaptation instances. |

968 ### 7.2.18 Adaptation: ScopesElement: CIM_ManagementSAP

969 Implementation of the ScopesElement adaptation is conditional.

970 Condition: The ScopedLaunchPoints (7.1.1) feature is implemented.

971 The ScopesElement adaptation models the relationship between a ScopedElement and a LaunchPoint.

972 Each ScopesElement instance shall have values that conform to the requirements in Table 33 and the
973 schema.

974 **Table 33 – ScopesElement: CIM_ManagementSAP element constraints**

| Elements | Requirement | Description |
|---|---|---|
| AvailableSAP | Mandatory | The value shall reference a LaunchPoint where the AvailableSAP.ManagementIsRestricted property is True. |
| ManagementIsRestricted | Mandatory | The value shall reference a ScopedElement that is a kind of a class named in an entry of the AvailableSAP.ManagedClasses property. |

975 ScopesElement associations shall not be instantiated if the ScopedLaunchPoints feature is not
976 implemented (see 7.1.1).

977 An instance of ScopesElement is required between a ScopedElement and each applicable LaunchPoint.

978 ### 7.2.19 Adaptation: ScopingProfile: CIM_RegisteredProfile

979 Implementation of the ScopingProfile adaptation is mandatory.

980 The ScopingProfile models an autonomous profile that incorporates this profile (as represented by the
981 LaunchProfile) as a component.

982 The ScopingProfile is associated directly or indirectly to each instance of the PublishingSystem
983 adaptation and to the LaunchProfile adaptation by a means specified by the profile of the ScopingProfile
984 and by the Profile Registration profile (DSP1033).

985 The profile represented by the ScopingProfile may further constrain the adaptations specified by this
986 profile.

987 There shall be an instance of the ScopingProfile adaptation.

# 8 Use cases

This clause has two primary types of clauses. The first clause describes a representative set of use cases from the perspective of the user of the primary management service. The remaining clauses describe use cases that the conformant managed system is required to support.

## 8.1 End user use cases

The use cases described in this clause are informative. They represent possible functionality provided by management applications that utilize managed systems that are conformant with this profile.

### 8.1.1 Background

A starting assumption is that a management application has a predefined set of features that it can offer its clients. Those features provide a level of management support for discovered resources.

The addition of this functionality enables an updated version of that application to discover additional features for the discovered resources. Those additional features are supported by other "auxiliary" management services. The management application can use that new information to provide its clients the ability to launch those other applications with context information.

The information required to invoke the auxiliary services is published by the managed systems as launch points. Regardless of where the auxiliary management applications are actually hosted, the launch information is published in a managed system as if the named auxiliary management application is remote to the managed system. It is expected that each managed system publish launch points for auxiliary management applications known to the vendor of the managed system. This information may or may not include information about management applications from other vendors.

### 8.1.2 PublishSpecifiedLaunchPoint

One means to extend the coverage is to use a management application with knowledge of another vendor’s managed system and with access to your managed system. That application would get information from the other vendor’s managed system and add them to yours using the ClientPublishing feature.

### 8.1.3 FederateLaunchPoints

A more dynamic means of extending coverage is to federate the information from the managed systems in the enterprise. In many cases, one managed system utilizes other managed systems. Often resources of the utilized systems are represented in the utilizing managed system. In such cases, the utilizing system can federate relevant launch point information from the utilized system by copying that information to the utilizing system. This information is then directly available to the management application of the utilizing system.

### 8.1.4 PublishDiscoveredLaunchPoints

The SMI-S Host Discovered Resources (HDR) profile models storage as seen by the operating system of one managed system and provides correlation to storage resource of underlying managed systems. An HDR client can see all the disk partitions, the relationship of partitions to the underlying disks, and the paths between HBA ports, device ports, and disk logical units. Non-disk logical units are also modeled. In some cases, disks seen from the operating system are actually array LUNs. And, in some of these cases, the array vendor has provided information through the data path (for example, SCSI VPD page 0x83, management URIs in fabric management services "platform" data structures, or a proprietary interface) that allow host-side components to determine the network address and path to a management application for the array.

The HDR-managed system can use this information – along with SCSI volume identifiers – to create a launch point for the array's element manager focused on the specified LUN. Of course this presumes prior agreement on how to form the URL. Such an agreement might be specified in the array profile. (For an example of the result, see Figure 10.)

The example in Figure 10 also shows how the generic functionality of an element manager can be specified. This might cause the welcome screen of the management application to be invoked.

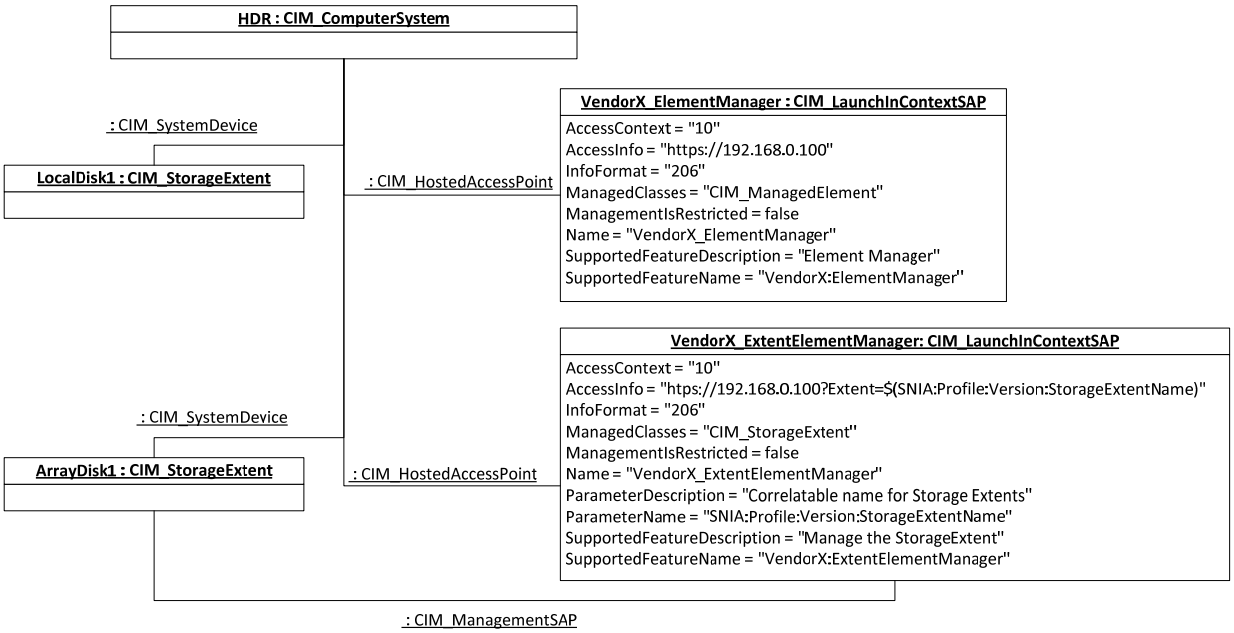



**Figure 10 – HDR example showing result of adding a launch point**

### 8.1.5 ProfileSpecifiedLaunchPoints

A profile may add a Launch in Context Definition table to describe the URL templates and parameters of a list of interoperable launch points that a client could call no matter who the vendor is and get to the same place. For example, FabricSwitchPortControl passing a Port WWN would go to the appropriate Switch Device Manager dialog to enable/disable a port so that the client could allow a right click on the port and see an option to do port control.

A similar approach could launch such things as the following:

- a tape library element manager focusing on a tape drive
- a volume management element manager focusing on a partition
- an HHRC element manager focusing on a LUN

## 8.2 Mandatory profile supported low-level use cases

The following use cases shall be supported by a conformant implementation of a managed system.

### 8.2.1 DiscoverConformance: Determine if the managed system advertises support for this profile

#### 8.2.1.1 Preconditions

The number of CIM_RegisteredProfile instances is not excessive. This would preclude using the simple enumeration used here.

#### 8.2.1.2 Flow of activities

The main sequence of activities is as follows:

1) Invoke the "Enumerate Profiles Advertised in Interop Namespace by an Implementation" use case as defined in DSP1033, but with the modification that the entire instance specification for each CIM_RegisteredProfile shall be returned.
2) Create a list of LaunchProfile adaptation instances by matching the returned CIM_RegisteredProfile instance specifications with the requirements of this specification (see 7.2.9).
3) Return the list.

#### 8.2.1.3 Postconditions

The system state is unchanged.

### 8.2.2 ListLaunchServices: Find LaunchService instances that conform to this profile

NOTE: This is the recommended process; however, current practice uses other discovery techniques, such as those defined in DSP1033.

#### 8.2.2.1 Preconditions

LaunchProfile instances are known (see 8.2.1). One such instance is used as a starting point.

#### 8.2.2.2 Flow of activities

The main sequence of activities is as follows:

1) Create an empty list of LaunchService instances.
2) Fill the list with conformant LaunchService instances.
   a) Execute the PullConformantInstance( ) method as specified in clause 7.2.12.3.
   b) Add the results that are central instances to the list.
   c) If EndOfSequence is false, invoke the PullConformantInstances( ) operation as specified in clause 7.2.12.3 and go to step 2.
   d) Invoke the CloseConformantInstances( ) operation as specified in clause 7.2.12.2.
3) Return the list.

#### 8.2.2.3 Postconditions

The system state is unchanged.

### 8.2.3 ListLaunchPoints: List advertised LaunchPoints

This is a primary means for management clients to discover auxiliary services that may be available for the elements known to a PublishingSystem.

#### 8.2.3.1 Preconditions

LaunchService instances are known (see 8.2.2). One such instance is used as a starting point.

#### 8.2.3.2 Flow of activities

The main sequence of activities is as follows:

1) Invoke the GetAssociatedInstancesWithPath( ) method as specified in 7.2.8.15 to return a list of LaunchPoint instances.
2) Return the list of LaunchPoint instances.

#### 8.2.3.3 Postconditions

The system state is unchanged.

### 8.2.4 AddLaunchPoint: Add a new LaunchPoint to the PublishingSystem

#### 8.2.4.1 Preconditions

The following pre-conditions must be met:

- The ClientPublishing feature is enabled.
- The client has picked a particular LaunchService.
- The client has identified an auxiliary service that provides additional features for classes of instances supported by the PublishingSystem. The client has created a URI template representing that auxiliary service, together with the set of parameters required to complete the template. The client has identified a (possibly empty) set of candidate Elements to which the new LaunchPoint is intended to be scoped.

#### 8.2.4.2 Flow of activities

The main sequence of activities is as follows:

1) Create an instance specification for a LaunchPoint (see7.2.8).
2) Using the above instance specification and the scoping list of instances, invoke LaunchService.CreateLaunchPoint as specified in 7.2.13.2.
3) Return the newly created LaunchPoint instance.

#### 8.2.4.3 Postconditions

The newly created LaunchPoint is instantiated and connected to the PublishingSystem, the LaunchService, and, if specified, a set of Elements.

### 8.2.5 RemoveLaunchPoint

A management client removes a LaunchPoint from the PublishingSystem.

#### 8.2.5.1 Preconditions

The following pre-conditions must be met:

- The ClientPublishing feature is enabled.
- The client has identified a particular LaunchPoint.

- The client has a null list of Elements if it has decided to completely remove that LaunchPoint or has a non-empty list of ScopedElement instance paths if it has decided only to remove those ScopedElements from the scope of the LaunchPoint.

#### 8.2.5.2 Flow of activities

The main sequence of activities is as follows:

1) Using the instance path to the identified LaunchPoint and the scoping list of ScopedElement instances, invoke LaunchService.RemoveLaunchPoint as specified in 7.2.13.3.

#### 8.2.5.3 Postconditions

If the scoping list of Elements was null, or if it named all scoped Elements, then the designated LaunchPoint instance is removed along with all associations to it. If the scoping list was not null and did not cover all scoped Elements, then only the ScopedElement associations are removed.

### 8.2.6 GetDerivedParameters: Ask PublishingSystem to return context-specific values for the parameters of a LaunchPoint-specified URI template

#### 8.2.6.1 Preconditions

The following pre-conditions must be met:

- The ParameterDerivation Feature is implemented.
- The management client has identified a particular LaunchPoint and has an instance path to a CIM_ManagedElement that represents the evaluation context.

#### 8.2.6.2 Flow of activities

The main sequence of activities is as follows:

1) Invoke LaunchPoint.GetDerivedParametersForElement as specified in 7.2.8.14.

#### 8.2.6.3 Postconditions

The ParameterValues output parameter contains an entry containing a parameter value for each parameter name in the ParameterName property (see 7.2.8.3).

# Annex A (normative)

# OCL Usage Guide

## A.1 Introduction

This profile takes advantage of two types of OCL expressions utilized as values of properties of CIM_LaunchInContext. The first is a derivation expression used in entries of the ParameterDerivation property. The second is an invariant expression used in entries of the ParameterConstraint property.

Each OCL statement is evaluated relative to a class. The ***self*** keyword refers to that class. If the class context represented by ***self*** is known in the context of evaluation, OCL allows it to be implied (see example in Table A-1).

Each OCL statement may contain expressions. These expressions are created using ***self***, various functions, and operators. The typical operators are evaluated in the following order (from first to last): not, - (negative), *, /, +, - (subtract), <, >, <=, >=, =, <>, and, or, and xor. Parentheses can be used to change the order of evaluation. Typical functions: ***sum(), count(),*** and ***like()*** are described in the clauses below.

## A.2 ParameterDerivation property

The general form of an OCL derivation constraint is as follows:

context className::propertyName: propertyType derive: <derivationExpression>

Only the *derivationExpression* is stored in the ParameterDerivation property.

As used by this feature, the derived value is returned in the corresponding ParameterValue entry returned by the GetDerivedParametersForElement( ) method of CIM_ExtendedLaunchInContextSAP. In other words, the *className::propertyName: propertyType* tokens correspond to the corresponding ParameterName and ParameterType properties.

If a non-empty value is returned in an entry of ParameterValue, it shall be formatted based on the corresponding ParameterType value and according to the rules defined for the ***defaultvalue*** production specified in Annex A ("MOF Syntax Grammar Description") of DSP0004.

For evaluation purposes, the ***self*** keyword refers to the class referenced by the ManagedElement parameter of the GetDerivedParametersForElement( ) method.

Consider the following simple example: GetDerivedParametersForElement is called with ManagedElement pointing to an instance of CIM_StorageExtent; the parameters and referenced property values are as specified in Table A-1.

**Table A-1 – ParameterDerivation example 1**

| Parameter Name | Parameter Derivation | Referenced Property Name | Referenced Property Value | Derived ParameterValue |
|---|---|---|---|---|
| StorageName | Name | Name or ***self.***Name | 21000020372D3C73 | 21000020372D3C73 |
| NameFormat | NameFormat | NameFormat | 9 | 9 |
| NameNamespace | NameNamespace | NameNamespace | 2 | 2 |

1176 These are simple examples of OCL expressions. The real power comes from the ability to follow
1177 associations, process regular expressions, and perform arithmetic.

1178 Consider the example shown in Table A-2, which uses the same instance of CIM_StorageExtent.

1179 **Table A-2 – ParameterDerivation example 2**

| ParameterName | ParameterDerivation | Referenced Property Name | Referenced Property Value | Derived ParameterValue |
|---|---|---|---|---|
| | | BlockSize | 512 | |
| | | ConsumableBlocks | 100 | |
| | | NumberOfBlocks | 110 | |
| FormatedBytes | ConsumableBlocks * BlockSize | | | 51200 |
| TotalBytes | NumberOfBlocks * BlockSize | | | 52320 |

1180 When traversing associations, keep the following points in mind:

1181 1) In the CIM architecture, all associations are represented by association classes. Because of the
1182 way they are defined and used, the association role names are not a reliable means to navigate
1183 over an association. Additionally, association classes may have properties that need to be
1184 accessed. The roles are also properties of the association class. For these reasons, navigation
1185 over an association always starts by naming the association class. Next, a property of the
1186 association class is named. Assuming that self refers to an instance of CIM_StoragePool,
1187 navigation to properties of CIM_AllocatedFromStoragePool looks like this:

1188 self.CIM_AllocatedFromStoragePool.propertyName

1189 or

1190 CIM_AllocatedFromStoragePool.propertyName

1191 The token propertyName represents a property of the association class. If that property is a
1192 string qualified as an EmbeddedInstance or if it is a reference property, then the type of the
1193 property is the embedded or referenced class. Otherwise, it has the native CIM type of the
1194 property. If no property is specified, then the type is that of the association class. Instances of
1195 the association class CIM_AllocatedFromStoragePool can be retrieved by:

1196 CIM_AllocatedFromStoragePool

1197 2) Whenever you navigate through an association, you are typically creating a collection (or array).
1198 The following rules apply:

1199 a) An association class instance is addressed as if it were an embedded property of each
1200 referenced class.

1201 – That property can be treated as a scalar association class instance if the multiplicity of
1202 the opposite role is defined as exactly one.

1203 – Otherwise, that property is treated as an array of association class instances. The
1204 cardinality of the array depends on the number of instances pointed to by the opposite
1205 end.

1206 b) The cardinality of each scalar property (including the roles) is one.

1207 c) The cardinality of array properties depends on the number of entries at the time of
1208 reference.

3) Array elements can have basic CIM types, or be references, embedded instances, or association class instances. The latter three entry types are all treated as structures containing properties defined by the referenced class, the class of the embedded instance, or the association class. In those cases, the properties of the structure are referenced using 'dot' notation.

4) OCL provides some useful built-in functions for collections of properties. These apply to collections created because the path to a property includes an association or because the named property is defined as an array. A word of caution: The array size is the product of the cardinalities of any associations involved and of cardinalities of the properties.

Table A-3 shows some examples of collection operations.

**Table A-3 – Example collection operations**

| Function | Description |
|---|---|
| propertyName->sum() | Returns the sum of the values of the named property |
| associationClass.role.<br>arrayPropertyName->count() | Returns the sum of the cardinality of each array property across all instances of associationClass |
| CIM_BasedOn.Antecedent.<br>NumberOfBlocks->sum() | Assuming self is a CIM_StorageExtent, this expression returns the sum of the blocks underlying the underlying storage extents. |

## A.3 ParameterConstraint property

The general form of an OCL invariant constraint is as follows:

**context** className::propertyName: propertyType **inv:** <invariantExpression>

Only the *invariantExpression* is stored in the ParameterConstraint property.

The assumption for the ParameterConstraint property is that the management client provides the parameter value. The constraint is on that value and is not written as a constraint on the managed system.

The OCL ***self*** keyword refers to the corresponding parameter value and has type expressed in the corresponding ParameterType entry. Because this is always a scalar expression and it has no related associations, an *invariantExpression* is typically quite simple.

Such expressions must evaluate to a boolean true or false. This may be a result of expression evaluation or as a result of a logical compare against two expressions.

The ***like*** boolean function is provided for string comparison; when used here, it would be expressed as:

*self*.**like(**regularExpression**)**

The rules for a regularExpression are defined in *XQuery 1.0 and XPath 2.0 Functions and Operators*,clause 7.6.1, "Regular Expression.Syntax".

Table A-4 shows example parameter constraints.

1237 **Table A-4 – Example parameter constraints**

| ParameterConstraint element contains: | The management client must assure that: |
|---|---|
| self >= 0 and self <= 5 | The value is valid if it is between 0 and 5. |
| self.like('^([0-9A-F]{1,4}:){7}[0-9A-F]{1,4}$') | The value conforms to an Internet Protocol version 6 address. |

1238

# Annex B
(informative)

# Change log

| Version | Date | Description |
|---|---|---|
| 1.0.0 | 2010-10-21 | Released as DMTF Standard |
| | | |

# Bibliography

DMTF DSP1004, *Base Server Profile* 1.0,
http://www.dmtf.org/standards/published_documents/DSP1004_1.0.pdf

OMG, *OCL 2.0, OMG Final Adopted Specificatio*n,
http://www.omg.org/spec/OCL/2.0