Date: September 2012

SES Management TelcoML Extension

FTF Beta 2

OMG Document Number: ptc/2012-09-23

Standard document URL:

http://www.omg.org/spec/TelcoML-SES/1.0/PDF

Associated File(s)*:

http://www.omg.org/spec/TelcoML-SES/20120924
(XMI normative – SES Management TelcoML Extension)

Source document: SES Management TelcoML Extension

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

NOTICES REGARDING SOFTWARE, DOCUMENTS AND SERVICES SUPPLIED BY THE TM FORUM. IN NO EVENT SHALL TM FORUM BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF SOFTWARE, DOCUMENTS, PROVISION OF OR FAILURE TO PROVIDE SERVICES, OR INFORMATION SUPPLIED BY TM FORUM.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 140 Kendrick Street, Needham, MA 02494, U.S.A.

TRADEMARKS

MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management Group™, OMG™ , Unified Modeling Language™, Model Driven Architecture Logo™, Model Driven Architecture Diagram™, CORBA logos™, XMI Logo™, CWM™, CWM Logo™, IIOP™ , MOF™ , OMG Interface Definition Language (IDL)™ , and OMG SysML™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the

specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page http://www.omg.org, under Documents, Report a Bug/Issue (http://www.omg.org/technology/agreement.)

Table of Contents

Figures

Preface

About the OMG Management Group

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable, and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies, and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at http://www.omg.org/.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. A Specifications Catalog is available from the OMG website at:

http://www.omg.org/technology/documents/spec_catalog.htm

Specifications within the Catalog are organized by the following categories:

OMG Modeling Specifications

- UML
- MOF
- XMI
- CWM
- Profile specifications

OMG Middleware Specifications

- CORBA/IIOP
- IDL/Language Mappings
- Specialized CORBA specifications
- CORBA Component Model (CCM)

Platform Specific Model and Interface Specifications

- CORBAservices
- CORBAfacilities
- OMG Domain specifications
- OMG Embedded Intelligence specifications
- OMG Security specifications

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format,

may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
140 Kendrick Street
Building A, Suite 300
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*

Certain OMG specifications are also available as ISO standards. Please consult *http://www.iso.org*

Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

Helvetica/Arial - 10 pt. Bold: OMG Interface Definition Language (OMG IDL) and syntax elements.

`Courier - 10 pt. Bold:` Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

NOTE: Terms that appear in italics are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

Issues

The reader is encouraged to report any technical or editing issues/problems with this specification to *http://www.omg.org/technology/agreement.htm*

1 Scope

The objective of this specification is to extend the TelcoML Enabler Library (see TelcoML specification) with the Software Enabled Services Management Interface (SES Management Interface/SMI) specified by the TeleManagement Forum. To that end the original SMI WSDL and XSD files have been retro-modeled to obtain the equivalent TelcoML representation.

NOTE: The SMI specification published by the TM Forum is the definitive version of the APIs from which the TelcoML SES Management Interfaces are derived. Refer to "DISCLAIMER OF WARRANTY" section for limited liability of using TM Forum specifications.

2 Conformance

This specification adds an orthogonal dimension to the two conformance levels defined by the core TelcoML specification (**TelcoML Editing** and **TelcoML Execution**). This dimension indicates whether the Enabler Library being supported includes the Software Enabled Services Management extension defined by this specification.

A UML tool that is **TelcoML Editing With Service Management** is a UML tool that is "TelcoML Editing" compliant and in which the supported enabler library includes the Software Enabled Services Management enabler.

A UML tool that is **TelcoML Execution With Service Management** is a UML tool that is "TelcoML Execution" compliant and in which the supported enabler library includes the Software Enabled Services Management enabler.

We recall below the two conformance levels from the core TelcoML specification.

- ***TelcoML Editing***: A UML tool that has pre-installed the model elements of the TelcoML profile: the interface definitions specified in the Enabler Library, the list of stereotypes of the Composition Profile. Such tool should provide means to edit state machines in line with TelcoML notation conventions defined by the Composition Profile. In addition such tool should provide export facility using either UML compliant XMI 2.1 or ALF representation.

- ***TelcoML Execution***: A UML tool that supports TelcoML Edition compliance level and that provides in addition means to execute or simulate service compositions specified using the TelcoML composition profile. This implies means to connect service interfaces in the Enabler Library to concrete implementations of these telecommunication facilities.

3 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

List of normative references.

1. *Object Constraint Language (OCL) v2.3.1*, OMG document number formal/2012-05-09.

2. *Service oriented architecture Modeling Language (SoaML) v1.0*, OMG document number formal/2012-05-01.

3. *UML v2.4.1 Superstructure (UML)*, OMG document number formal/2012-05-07.

4. *Action Language for Foundational UML (ALF)*, OMG document number ptc/2010-10-05.

5. Meta Object Facility (*MOF) Core v2.4.1*, OMG document number formal/2011-08-07.

6. *MOF/XMI Mapping v2.4.1*, OMG document number formal/2011-08-09.

Normative parts

The following are included as part of this specification.

- XMI Document for SES Management TelcoML Extension mars/2012-09-24

4 Terms and Definitions

This specification does not introduce specific definitions and terms.

5 Symbols

List of symbols/abbreviations.

UML: Unified Modeling Language

SoaML: Service Oriented Modeling Language

XSD: XML Schema Definition

TelcoML: Telecommunication Modeling Library

SES : Software Enabled Service

SMI: Software Enabled Services Management Interface

6 Additional Information

6.1 Acknowledgements

The following persons were main responsible for the specification:

Antonio Cruz (Portugal Telecom), Eric Troup (Microsoft), Johan Vandenberghe (Alcatel-Lucent), Mariano Belaunde, Hannebelle Jérôme (France Telecom/Orange), Irv Badr (IBM), Jenny Huang (AT&T)

The following companies submitted this specification:

- International Business Machines
- Unisys

The following companies supported this specification:

- France Telecom - Orange Labs
- AT&T
- Portugal Telecom
- Microsoft
- European Software Institute
- DoCoMo Communication Laboratory Europe GmbH
- SINTEF
- Telefonica

7 Software Enabled Services Management Overview

This section is non-normative.

7.1 Context

The Software Enabled Services (SES) Management Solution provides a means to allow consistent end-to-end management and metering of services exposed by and across different service providers domains and technologies, such as communication or Web 2.0 services. In concrete terms it delivers the ability to configure, activate or suspend a service instance execution and to both receive or being notified of any kind of metrics, health state and detailed information about eventual failures.

The following operations are typically offered to manage a SES:

- Activation of a SES: Making the SES available for a particular context (Deploying the SES)
- Provisioning of a SES: Configuring the settings of a SES or a SES instance
- State monitoring of a SES: Querying the history and current status in terms of life cycle management (for a specific instance of the SES) and our listening for status updates
- Usage monitoring of a SES: Querying for usage metrics from the SES instance or listening for usage metrics reports or alarm (e.g. if metrics conditions imply notifications)
- Health monitoring of a SES: Querying for health metrics from the SES instance or listening to alarm from the resource
- Update of a SES: Modification of the setting or life cycle management status of a SES instance
- De-activation of a SES: making the SES unavailable in a particular context.

More information on Software Enabled Service (SES) Management initiative can be found at:
http://www.tmforum.org/SoftwareEnabledServices/4664/home.html

and
http://www.tmforum.org/InformationAgreements/TMF617SoftwareEnabled/48632/article.html

7.2 TelcoML Extension for Software Enabled Services Management

The TelcoML Enabler library defines a set of predefined service interfaces to facilitate the definition of composite services accessing well-known telecommunication facilities (called *enablers*) usually provided by telecom operators. These service definitions are provided in a sufficient technology agnostic way to enable independence in respect to the enabler provider and to the usage of a specific implementation technology.

In section 8 we specify formally the extension to the TelcoML Enabler Library for Software Enabled Services Management. It consist of the addition of one service enabler defining two interfaces: SMI (Software Enabled Services Management Interface) is the provider interface and SMIManagementReportNotification is the interface to be implemented by a consumer to receive notifications. Accompanying these two interfaces we defined a list of data types involved in the usage of the interfaces. This enabler has been originally specified by the Tele-Management Forum as part of the SES/SMI initiative. This specification reformulates the interface using TelcoML conventions.

8 Software Enabled Services Management Enabler

This section is normative.

The Software Enabled Services (SES) Management enabler provides to consumers the *SMI* interface with simple management operations to manage and control the execution of running services instances. This includes the ability for the consumer to be notified of changes in the execution. To emit these notifications the consumer implements the notification interface named *SMIManagementReportNotification*. The relationship between these two interfaces is depicted below.

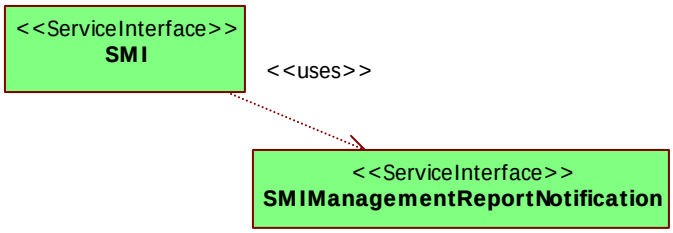

Figure 1: Interfaces for Software Enabled Services Management

In the following sub-sections we describe these two interfaces.

8.1 SMI Interface

8.1.1 Overview and diagrams

The SMI supports the following operations to allow SES components to interact with management systems in a consistent way.

• getExecutionState returns the current execution state of a service instance;
• getManagementReport returns a report containing information about the service instance health, execution state, eventual failures and metrics (usage, performance for example);
• getServiceConfiguration returns data that describe the current set configuration values used by the service instance;
• setExecutionState allows a service consumer to activate or suspend service execution;
• setServiceConfiguration applies configuration values used by the service instance;
• registerListener sets the communication endpoint address to enable emitting notifications to consumers;
• unregisterListener de-activates the notification mechanism.

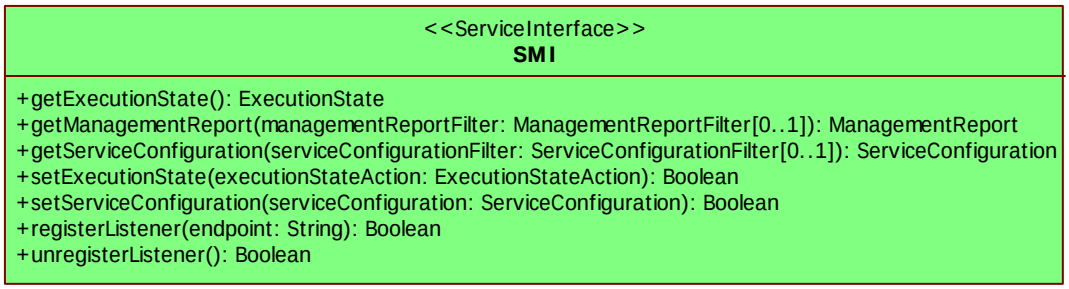

Figure 2: SMI interface

The specification of the SMI interface involves the definition of various data types are depicted in Figure 3.

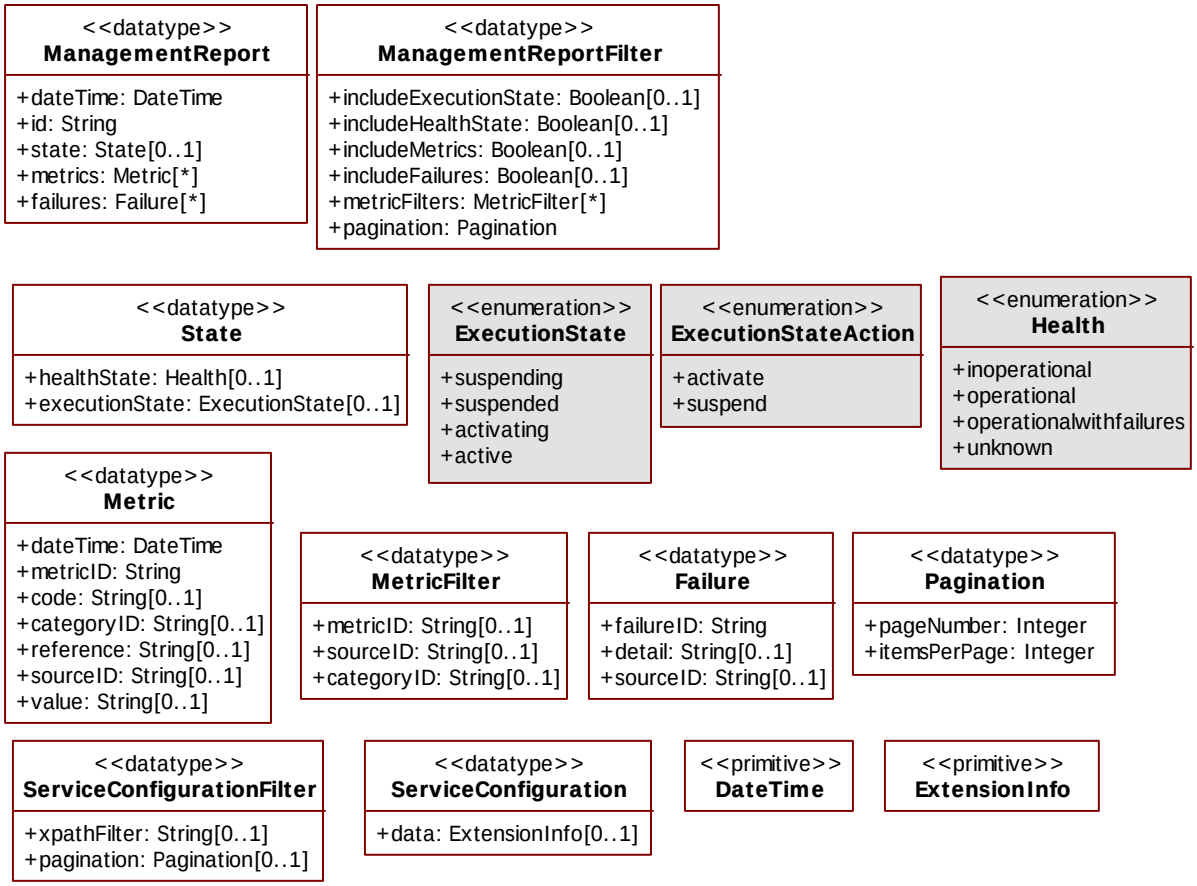

Figure 3: Specific data types for the SMI Interface

8.1.2 Structured Data Types

8.1.2.1 Failure

A *failure* represents an error raised during the execution of a given service.

Attributes:

- *failureID: String[1]* A string used for identifying a particular kind of failure. This identifier maybe defined in a global (per organization or per domain) list of failure codes.

- *detail: String[0..1]* A detailed description of the failure found. An example of this fields content could be the stack trace of a code exception raised at runtime.

- *sourceID: String[0..1]* A unique identifier that indicates the service or resource causing the failure. The source could be the service itself or another service being composed or a resource being used. This identifier could then be used to locate the service or resource in a centralized list or graph of services and resources that may exist per organization or domain.

8.1.2.2 ManagementReport

A *management report* contains information about the service instance health, execution state, eventual failures and metrics (usage, performance for example).

Attributes:

- *dateTime: DateTime[1]* The date time the report is generated.

- *id*:*String[1]* An identifier for the report (unique).

- *state*:*State[0..1]* The state of the requested service instance.

- *metrics*:*Metric[*]* The metrics computed on the service instance.

- *failures :Failure[*]* The failures regarding the activation and the execution of the service instance.

8.1.2.3 ManagementReportFilter

A *management report filter* allows the service consumer to select which information will be returned when requesting a management report and also to control the amount of *metrics* returned, using a *pagination* element included the filter data.

Attributes:

- *includeExecutionState: Boolean[0..1]* Flag to control if the execution state should be reported.

- *includeHealthState: Boolean[0..1]* Flag to control if the health state should be reported.

- *includeMetrics: Boolean[0..1]* Flag to control if metric evaluation should be reported.

- *includeFailures: Boolean[0..1]* Flag to control if failure information should be reported.

- *metricFilters: MetricFilter[*]* Indicates the metrics to be included in the management report.

- *pagination: Pagination[1]* Controls the amount of metrics returned.

8.1.2.4 Metric

A *metric* represents a measure of a specific aspect of the performance of a service.

Attributes:

- *code: String[0..1]* A string used as a code for identifying a particular Metric in a list of metrics. This list can be available per organization or domain. It may also be centrally available a complete Metric template and metadata.

- *categoryID: String[0..1]* A String intended to be used to filter Metrics that belong to a given category, when using MetricFilter.

- *dateTime: DateTime[1]* the date time of the measure.

- *reference: String[0..1]* A string used to correlate the metric being reported with a particular service consumer or operation context.

- *sourceID: String[0..1]* An identifier that can be used to relate a metric with a particular service or resource that the reporting service instance depends on. A centralized list or graph of services and resources may exist.

- *value: String[0..1]* A value measured that can be formatted as an integer, a decimal or a percentage number, depending on the specific metric being represented.

- *metricID: String[1]* A unique identifying string for this metric information.

8.1.2.5 MetricFilter

A *metric filter* is part of a management report filter. It provides criteria for selecting the metrics that will be included in a management report.

Attributes:

- *metricID: String[0..1]* Contains the identifier of the metric that should be selected. This is optional since filtering can operate based on other criteria.

- *sourceID: String[0..1]* Provides the source identifier to be used for filtering. A source identifier is used to relate a metric with a particular service or resource that the reporting service instance depends on.

- *categoryID: String[0..1]* The category of the metric that should be used to filter metrics.

Constraints:

- One of *sourceID* or *metricID* attributes should be provided (but not both).

```
self.sourceID->notEmpty() xor self.metricID->notEmpty()
```

8.1.2.6 Pagination

A *pagination* data is used to indicate the amount of data items to be retrieved. This is done by providing the number of pages and the number of items per page.

Attributes:

- *pageNumber: Integer[1]* The number of pages to be retrieved.

- *itemsPerPage: Integer[1]* The number of items per retrieved page.

8.1.2.7 ServiceConfiguration

A *service configuration* describes the current set of configuration values used by the service instance. The structure of this data type depends on vendor specificities.

Note: Vendors will typically provide an XSD schema to describe the structure and reference it in an XML based serialization. Alternatively vendors may subclass ServiceConfiguration to define the actual data structure.
Attributes:

- *data: ExtensionInfo[0..1]* The vendor specific data to describe the set of configuration values.

8.1.2.8 ServiceConfigurationFilter

A *service configuration filter* allows the service consumer to select which information will be returned and also to control the amount of configuration values returned, using a *pagination* element.

Attributes:

- *xpathFilter: String[0..1]* The xpath expression to retrieve the selected configuration data.

- *pagination: Pagination[0..1]* Information to control the amount of configuration data to retrieve.

8.1.2.9 State

A *state* data type represents the current state of a running instance of a service, including its health condition and eventual failures (we assume the service is already in the operation phase of its lifecycle).

Attributes:

- *healthState: Health[0..1]* The health state of the current running service instance. Possible values are: *inoperational*, *operational*, *operationalwithfailures* and *unknown* (see Health enumeration).

- *executionState: ExecutionState[0..1]* The state of the running service instance. Possible values are: *suspending*, *suspended*, *activating* and *active* (see ExecutionState enumeration).

8.1.3 Enumerations

8.1.3.1 ExecutionState

The *execution state* enumeration type lists all possible values for the state of a running service instance.

Enumeration Literals:

- *activating* A service is starting or finalizing steps needed in order to become available.

- *active* A service is available to perform its capabilities.

- *suspending* A service is in the process of becoming unavailable to its consumers. An example of entering this state is when a service is finalizing its currently executing requests, after a Suspend request.

- *suspended* A service was made unavailable to its consumers. An example of this is the result state of suspending a service for maintenance.

8.1.3.2 ExecutionStateAction

The *execution state action* enumeration type lists all possible actions to change the execution state of a service instance.

Enumeration Literals:

- *activate* The action that makes the service available to perform its capabilities.

- *suspend* The action to temporarily stops the execution of the service.

8.1.3.3 Health

The *health* enumeration type lists all possible health conditions for a service instance.

Enumeration Literals:

- *inoperational* The service is not ready to consumers.

- *operational* The service expected behavior is guaranteed to consumers.

- *operationalwithfailures* The service expected behavior cant be guaranteed to consumers.

- *unknown* The current health state of the service could not be determined.

8.1.4 Primitive Types

8.1.4.1 DateTime

A *date time* is a primitive type used to represent dates and times.
The general syntax is *CCYY-MM-DDThh:mm:ss.sss*. Example: 2004-04-12T13:20:00.
Note: DateTime corresponds to the XSD *dateTime* builtin type
(more information available at: http://www.schemacentral.com/sc/xsd11/t-xsd_dateTime.html).

8.1.4.2 ExtensionInfo

An *extension info* is a primitive type used to represent vendor specific data. This is an opaque data type meaning that it may potentially have any structure.

Note: A serialization in XML of this data type will typically exploit the XSD *any* built-in element (see http://www.schemacentral.com/sc/xsd11/e-xsd_any.html).

8.1.5 Operations

8.1.5.1 Retrieving the execution state

getExecutionState(): ExecutionState

Returns the current *execution state* of a service instance. The possible states are: *activating*, *active*, *suspending* and *suspended*.

Parameters: None

Outputs:

An *ExecutionState*, describing the current execution state of a service instance. Possible values are *activating*, *active*, *suspending* and *suspended.*

Exceptions:

AccessDenied : Raised when the operation fails for security reasons.
CommunicationLoss: The service is unable to communicate with an underlying system or resource, and such communication is required to complete the request.
InternalError : The request has resulted in an internal error.
InvalidInput: Raised for all failures related to operation input parameters.
NotImplemented: Raised if the entire request is not supported or the request with the specified input parameters is not supported.
UnableToComply: Raised if cannot respond to the request (general case).

8.1.5.2 Retrieving the management report

getManagementReport(managementReportFilter: ManagementReportFilter[0..1]) : ManagementReport

Returns a *management report* containing information about the service instance health, execution state, eventual failures and metrics (usage, performance for example). Optionally, it accepts a *management report filter* input parameter that allows the service consumer to select which information will be returned and also to control the amount of metrics returned, using a *pagination* element in the filter data.

The operation will return a complete management report when no filter is provided.

Parameters:

managementReportFilter:ManagementReportFilter[0..1] : The optional filter used to select the information to be returned.

Outputs:

A *ManagementReport*, containing information on actual execution of a service instance.

Exceptions:

AccessDenied : Raised when the operation fails for security reasons.
CommunicationLoss: The service is unable to communicate with an underlying system or resource, and such communication is required to complete the request.
InternalError : The request has resulted in an internal error.
InvalidInput: Raised for all failures related to operation input parameters.

NotImplemented: Raised if the entire request is not supported or the request with the specified input parameters is not supported.
UnableToComply: Raised if cannot respond to the request (general case).

8.1.5.3 Retrieving the service configuration

```
getServiceConfiguration(
  serviceConfigurationFilter: ServiceConfigurationFilter[0..1])
  : ServiceConfiguration
```

Returns the *service configuration* describing the current set of configuration values used by the service instance. Optionally, it accepts a *service configuration filter* input parameter that allows the service consumer to select which information will be returned and also to control the amount of configuration values returned, using a *pagination* element included in the filter data.

The operation returns a complete service configuration when no filter is provided.

Parameters:

serviceConfigurationFilter: *ServiceConfigurationFilter[0..1]* The optional filter to select what information should be returned.

Outputs:

A *ServiceConfiguration*, reflecting the configuration parameters of the service.

Exceptions:
AccessDenied : Raised when the operation fails for security reasons.
CommunicationLoss: The service is unable to communicate with an underlying system or resource, and such communication is required to complete the request.
InternalError : The request has resulted in an internal error.
InvalidInput: Raised for all failures related to operation input parameters.
NotImplemented: Raised if the entire request is not supported or the request with the specified input parameters is not supported.
UnableToComply: Raised if cannot respond to the request (general case).

8.1.5.4 Changing the execution state

```
setExecutionState(executionStateAction: ExecutionStateAction): Boolean
```

Allows a service consumer to activate or suspend service execution. It returns *true* if the change of service execution state requested by the consumer was made successfully. It has one input parameter to decide weather to activate or to suspend the service instance.

Parameters:

executionStateAction: ExecutionStateAction[1] : The destination address that will receive the instant message.

Outputs:

A *Boolean* value. True if the operation succeeds, false otherwise.

Exceptions:
AccessDenied : Raised when the operation fails for security reasons.
CommunicationLoss: The service is unable to communicate with an underlying system or resource, and such communication is required to complete the request.
InternalError : The request has resulted in an internal error.
InvalidInput: Raised for all failures related to operation input parameters.
NotImplemented: Raised if the entire request is not supported or the request with the specified input parameters is not supported.
UnableToComply: Raised if cannot respond to the request (general case).

8.1.5.5 Changing the service configuration

`setServiceConfiguration(serviceConfiguration: ServiceConfiguration): Boolean`

Applies configuration values used by the service instance. It requires a *service configuration* input parameter that describes the configuration values to be applied to the service instance.

Parameters:

serviceConfiguration: ServiceConfiguration[1] : Contains the configuration values to be applied to the service instance.

Outputs:

A *Boolean* value. *True* if the listener was successfully unregistered, *false* otherwise.

Exceptions:

AccessDenied : Raised when the operation fails for security reasons.
CommunicationLoss: The service is unable to communicate with an underlying system or resource, and such communication is required to complete the request.
InternalError : The request has resulted in an internal error.
InvalidInput: Raised for all failures related to operation input parameters.
NotImplemented: Raised if the entire request is not supported or the request with the specified input parameters is not supported.
UnableToComply: Raised if cannot respond to the request (general case).

8.1.5.6 Registering to receive notifications

`registerListener(endpoint: String): Boolean`

Sets the communication endpoint address the service instance must use to deliver a report containing information about their health, execution state, eventual failures and metrics. It returns *true* if the registration is successful. Note that filtering of the notifications is responsibility of the SMI consumer.

Parameters:

endpoint:String The address to be used for delivering the notification to the consumer.

Outputs:

A *Boolean* value. Returns true to indicate whether the listener was successfully registered, *false* otherwise.

Exceptions:

AccessDenied : Raised when the operation fails for security reasons.
CommunicationLoss: The service is unable to communicate with an underlying system or resource, and such communication is required to complete the request.
InternalError : The request has resulted in an internal error.
InvalidInput: Raised for all failures related to operation input parameters.
NotImplemented: Raised if the entire request is not supported or the request with the specified input parameters is not supported.
UnableToComply: Raised if cannot respond to the request (general case)..

8.1.5.7 Unregistering to notifications

`unregisterListener(): Boolean`

Clears the communication endpoint address that was previously set by calling the *registerListener* operation. It has no input operation. It returns *true* if the operation succeeds.
A second and subsequent *registerListener* calls should be rejected by the service as it does not support multiple listeners. First an *unregisterListener* needs to be called before a new *registerListener* can be triggered.

Parameters:

None.

Outputs:

A *Boolean* value. Returns *true* to indicate whether the listener was successfully unregistered, *false* otherwise.

Exceptions:

AccessDenied : Raised when the operation fails for security reasons.
CommunicationLoss: The service is unable to communicate with an underlying system or resource, and such communication is required to complete the request.
InternalError : The request has resulted in an internal error.
InvalidInput: Raised for all failures related to operation input parameters.
NotImplemented: Raised if the entire request is not supported or the request with the specified input parameters is not supported.
UnableToComply: Raised if cannot respond to the request (general case).

8.2 SMIManagementReportNotification Interface

8.2.1 Overview and diagrams

The SMIManagementReportNotification interface is called by an SMI implementation to notify a consumer on events related to the management of a running service instance (based on the endpoint address configured at subscription).

Figure 4: Interface for consumers to receive SMI notifications

The structure of the notification event is defined by the *ManagementReportNotification* data type.

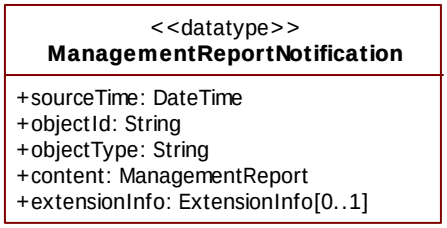

Figure 5: Event data structure in notifications

8.2.2 Data Types

8.2.2.1 ManagementReportNotification

A *management report notification* represents the event data generated as the result of a change in the state (execution or health state) of a service. This contains metrics and failures.

Attributes:

- *sourceTime: DateTime[1]* The time at which the event was reported by the source system.

- *objectId: String[1]* The identifier of the object associated with the event, as internal opaque identifier.

- *objectType: String[1]* The type (class) of the object associated with the event. This attribute is needed to allow simple notification filtering based on the object type.

- *content: ManagementReport[1]* The management report being notified to the consumer (see the ManagementReport data type definition in SMI Structured Data Types clause).

- *extensionInfo: ExtensionInfo[0..1]* Used for vendor extensions.

8.2.3 Operations

`acceptNotification(notification:ManagementReportNotification)`

Operation called by the system implementing the SMI interface to deliver a management report to the consumer. The structure of the notification is specified by the *management report notification* data type.

Parameters:

> *notification:ManagementReportNotification* The notification information containing the management report and possible other vendor extensions.

Outputs:

> None.

Exceptions:

> None.

Annex A: Textual Definition

(non normative)

We provide below a non-normative human-readable textual representation of the enabler specification. Note that the normative specification is provided in XMI in a separated accompanying document.

```
servicelibrary SoftwareEnabledServicesManagementLibrary {
servicepackage SoftwareEnabledServicesManagement {

service SMI {
  operation getExecutionState(): ExecutionState;
  operation getManagementReport(
    managementReportFilter: ManagementReportFilter[0..1]) : ManagementReport;
  operation getServiceConfiguration(
    serviceConfigurationFilter: ServiceConfigurationFilter[0..1]): ServiceConfiguration;
  operation setExecutionState(executionStateAction: ExecutionStateAction): Boolean;
  operation setServiceConfiguration(serviceConfiguration: ServiceConfiguration): Boolean;
  operation registerListener(endpoint: String): Boolean;
  operation unregisterListener(): Boolean;
}

service SMIManagementReportNotification {
  operation acceptNotification(notification: ManagementReportNotification);
}

primitive DateTime;
primitive ExtensionInfo;

enumeration ExecutionState {
  suspending,
  suspended,
  activating,
  active
}

enumeration ExecutionStateAction {
  activate,
  suspend
}

enumeration Health {
  inoperational,
  operational,
  operationalwithfailures,
  unknown
}

datatype ManagementReport {
  dateTime: DateTime;
  id: String;
  state: State[0..1];
  metrics: Metric[*];
  failures: Failure[*]
}
```

```
datatype ManagementReportFilter {
  includeExecutionState: Boolean[0..1];
  includeHealthState: Boolean[0..1];
  includeMetrics: Boolean[0..1];
  includeFailures: Boolean[0..1];
  metricFilters: MetricFilter[*];
  pagination: Pagination;
}

datatype State {
  healthState: Health[0..1];
  executionState: ExecutionState[0..1];
}

datatype Metric {
  dateTime: DateTime;
  metricID: String;
  code: String[0..1];
  categoryID: String[0..1];
  reference: String[0..1];
  sourceID: String[0..1];
  value: String[0..1];
}

datatype MetricFilter {
  metricID: String[0..1];
  sourceID: String[0..1];
  categoryID: String[0..1];
}

datatype Failure {
  failureID: String;
  detail: String[0..1];
  sourceID: String[0..1];
}

datatype Pagination {
  pageNumber: Integer;
  itemsPerPage: Integer;
}

datatype ServiceConfiguration {
  data : ExtensionInfo[0..1];
}

datatype ServiceConfigurationFilter {
  xpathFilter: String[0..1];
  pagination: Pagination[0..1];
}


datatype ManagementReportNotification {
  sourceTime: DateTime;
  objectId: String;
  objectType: String;
  content: ManagementReport;
  extensionInfo: ExtensionInfo[0..1];
}

}
}
```