Unified Service Description Language (USDL) Service Level (SLA) Module

February 28, 2011
Last Changed: May 24, 2011

Abstract. This document describes the Service Level Module in the third version of the Unified Service Description Language (USDL). USDL was developed as a holistic approach to describe entities provisioned into service networks; an approach, which considers and connects business, operational (functional) and technical aspects of service description. The Service Level Module covers concepts around service level agreements (SLAs), i.e., the specification of agreed conditions for a service delivery, namely guaranteed states and guaranteed actions. The module is intended to cover the generic core concepts that apply to arbitrary domains and to be extensible towards specific domains and their service level needs.

Table of Contents

Acknowledgements

The information in this document details a publicly released specification of parts of the Unified Service Description Language (USDL) for dissemination and further exploitation through the Internet of Services Community in accordance with the terms set forth below. This document does not represent any prior commitment for standardization or implementation of any portion of this specification by SAP.

The information in this document was developed through the following publicly co-funded research projects THESEUS/TEXO, Australian Smart Services CRC, Premium Services, SLA@SOI.

THESEUS/TEXO is research project funded by the German Federal Ministry for Economics and Technology.

Premium Services is research project funded by the German Federal Ministry for Education and Research.

Australian Smart Services CRC is research and development partnership funded by the private sector and governments under the Australian Government's Cooperative Research Centre program.

SLA@SOI is a research project funded by the European Commission under the 7th Framework Programme.

The contributing authors are: Alistair Barros (SAP), Christian Baumann (SAP), Anis Charfi (SAP), Jan Finzen (Fraunhofer IAO[1]), Matthias Fluegge (Fraunhofer FOKUS[2]), Steffen Heinzl (SAP), Tom Kiemes (SAP), Uwe Kylau (SAP), Florian Marienfeld (Fraunhofer FOKUS), Norman May (SAP), Oliver Müller (SAP, ERCIS Münster[3]), Francesco Novelli (SAP), Daniel Oberle (SAP), Jonas Pattberg (Fraunhofer FOKUS), Philip Robinson (SAP), Benjamin Schmeling (SAP), Wolfgang Theilmann (SAP), Heiko Witteborg (SAP).

[1] Fraunhofer-Institut für Arbeitswirtschaft und Organisation (Fraunhofer Institute for Industrial Engineering and Organisation)
[2] Fraunhofer-Institut für Offene Kommunikationssysteme (Fraunhofer Institute for Open Communications Systems)
[3] European Research Center for Information Systems at the Westfälische Wilhelms-Universität Münster

Terms of Use Agreement

IMPORTANT- PLEASE READ CAREFULLY: THIS TERMS OF USE AGREEMENT ("AGREEMENT") IS A BINDING AGREEMENT BETWEEN SAP AG, A GERMAN COMPANY WITH OFFICES AT DIETMAR-HOPP-ALLEE 16, 69190 WALLDORF, GERMANY ("SAP"), AND YOU, BEING EITHER AN INDIVIDUAL OR SINGLE LEGAL ENTITY ("YOU" OR "YOUR") REGARDING YOUR USE OF THIS DOCUMENT AND ANY INFORMATION CONTAINED THEREIN ("MATERIAL"):

BY DOWNLOADING, COPYING, OR OTHERWISE USING THE MATERIAL, YOU AGREE TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, DO NOT DOWNLOAD, COPY, OR USE THE MATERIAL.

IF YOU ARE DOWNLOADING, COPYING, OR USING THE MATERIAL ON BEHALF OF YOUR EMPLOYER OR A S A CONSULTANT OR AGENT OF A THIRD PARTY (COLLECTIVELY "YOUR COMPANY"), YOU REPRESENT AND WARRANT THAT YOU HAVE THE AUTHORITY TO ACT ON BEHALF OF AND BIND YOUR COMPANY TO THE TERMS OF THIS AGREEMENT AND ANY REFERENCE TO YOU OR YOUR SHALL ALSO INCLUDE YOUR COMPANY.

1. "SAP ENTITY/ENTITIES" shall mean SAP's affiliates and its subsidiaries, defined as corporations or other entities of which SAP owns, either directly or indirectly, more than fifty percent (50%) of the stock or other equity interests.

   "SAP SOFTWARE" shall mean the software products of SAP and/or SAP ENTITIES marketed and licensed by SAP and/or SAP ENTITIES.

   "INTELLECTUAL PROPERTY RIGHTS" means patents of any type, design rights, utility models or other similar invention rights, copyrights, trademarks, service marks, trade secret or confidentiality rights, and any other intangible property rights including applications for any of the foregoing, in any country, arising under statutory or common law or by contract and whether or not perfected, now existing or hereafter filed, issued, or acquired.

2. SAP grants YOU a nonexclusive, royalty-free, fully paid up, worldwide right to use, copy, display, perform, transmit, translate and distribute the MATERIAL provided hereunder. This shall include the right to reproduce, adapt, modify and to create derivative works of the MATERIAL and to make, have made, offer to sell, sell, lease, or otherwise distribute any product, and to practice any method, embodying such MATERIAL (including the right to sublicense any of the foregoing rights).

   SAP reserves the right to modify, change or discontinue the MATERIAL without notice at any time.

   YOU must not remove, overprint or deface any notice of copyright, trademark, logo, legend, or other notice of ownership from any originals or copies of the MATERIAL accessed hereunder. YOU agree to comply with the terms of the SAP Copyright Policy and those terms found by clicking on Copyright/Trademark at the web page http://www.internet-of-services.com.

   FOR AVOIDANCE OF DOUBT, NOTHING IN THIS AGREEMENT SHALL BE DEEMED TO

   (I) TO ASSUME OR PROVIDE FOR THE TRANSFER OF OWNERSHIP OF ANY INTELLECTUAL PROPERTY RIGHTS. ALL INTELLECTUAL PROPERTY RIGHTS INCLUDED, WITHOUT LIMITATION, COPYRIGHT IN ANY MATERIAL PROVIDED HEREUNDER, SHALL VEST IN AND AT ALL TIMES REMAIN VESTED IN THE ORIGINATOR OF THAT INTELLECTUAL PROPERTY RIGHT.

   (II) GIVE YOU THE RIGHT TO MODIFY, COPY, DISTRIBUTE, TRANSMIT, DISPLAY, PERFORM, REPRODUCE, PUBLISH, LICENSE, CREATE DERIVATIVE WORKS FROM,

TRANSFER; OR SELL ANY SAP SOFTWARE OR OTHER PRODUCT FOR ANY REASON UNLESS OTHERWISE PERMITTED BY SAP.

3. Any MATERIAL made available hereunder is provided to YOU "as is". SAP does not guarantee or warrant any features or qualities of the MATERIAL or give any undertaking with regard to any other quality. No warranty or undertaking shall be implied by YOU from any published MATERIAL except to the extent SAP has expressly confirmed such warranty or undertaking in writing. Warranties are validly given only with the express written confirmation of SAPs management.

   SAP does not represent or endorse the accuracy or reliability of any MATERIAL provided hereunder. SAP shall not be liable for damages caused by the use of the MATERIAL, unless such damages have been caused by SAPs willful misconduct.

   TO THE EXTENT ALLOWABLE BY APPLICABLE LAW, SAP AND ITS AFFILIATES, SUBSIDIARIES, OFFICERS, EMPLOYEES, AGENTS, PARTNERS, AND LICENSORS ARE NOT LIABLE TO YOU FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, PUNITIVE, CONSEQUENTIAL, OR EXEMPLARY DAMAGES, INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR LOSS OF PROFITS, REVENUE, GOODWILL, USE, DATA, OR OTHER INTANGIBLE LOSSES (EVEN IF SAP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES), HOWEVER CAUSED, WHETHER IN CONTRACT, TORT, OR OTHERWISE, ARISING OUT OF OR RESULTING FROM: (i) THE USE OR THE INABILITY TO USE THE MATERIAL; (ii) THE COST OF PROCUREMENT OF SUBSTITUTE GOODS AND SERVICES ARISING OUT OF YOUR USE OR INABILITY TO USE THE MATERIAL; OR (iii) ANY OTHER MATTER RELATING TO THE MATERIAL PROVIDED HEREUNDER. NOTWITHSTANDING ANYTHING TO THE CONTRARY HEREIN, THESE LIMITATIONS SHALL NOT APPLY IN CASE OF INTENT BY SAP AND IN CASE OF SAPS STATUTORY LIABILITY FOR PERSONAL INJURY AND DEFECTIVE PRODUCTS.

4. This AGREEMENT represents the complete and full agreement. No verbal side-agreements exist. Any changes to this AGREEMENT must be made in writing. This applies also to the revocation of the requirements for the written form.
5. This AGREEMENT shall be governed by the laws of Germany. The sole place of jurisdiction for all disputes arising directly in connection with this AGREEMENT shall be Karlsruhe, Germany.

1 Introduction

As outlined in the central document of this series *"USDL Overview"*, services are becoming the backbone for electronic commerce. Especially the trend to provision IT-based services outside company "firewalls" with the help of intermediaries is on the increase, as it allows organizations to take new opportunities relatively quickly. In this context services are seen as tradable entities that constitute a well-defined, encapsulated, reusable and business-aligned set of capabilities. The term business service is used for such services, in order to distinguish them from other types, e.g., those that are provided in a service-oriented IT infrastructure within an organization.

The Unified Service Description Language (USDL) defines a way to describe services from a business and operational point of view and align this with the technical perspective. While the latter is captured quite well by existing service description languages, USDL explicitly enables to express business characteristics set by an organization. Their purpose is to provide means for consumers to invoke and use business services, and for intermediaries to (re)use and repurpose services. A detailed explanation of the scope and objectives of USDL is given in *"USDL Overview"*.

USDL on a whole is made up of a set of modules, each addressing different aspects of the overall service description. Modularization was introduced to improve readability of the model, which drastically grew in size compared to its predecessor. The modules have dependencies among each other (shown in Figure 1), as they may reuse concepts from other modules. Currently, there are 9 modules in the set that constitutes USDL version 3.0.

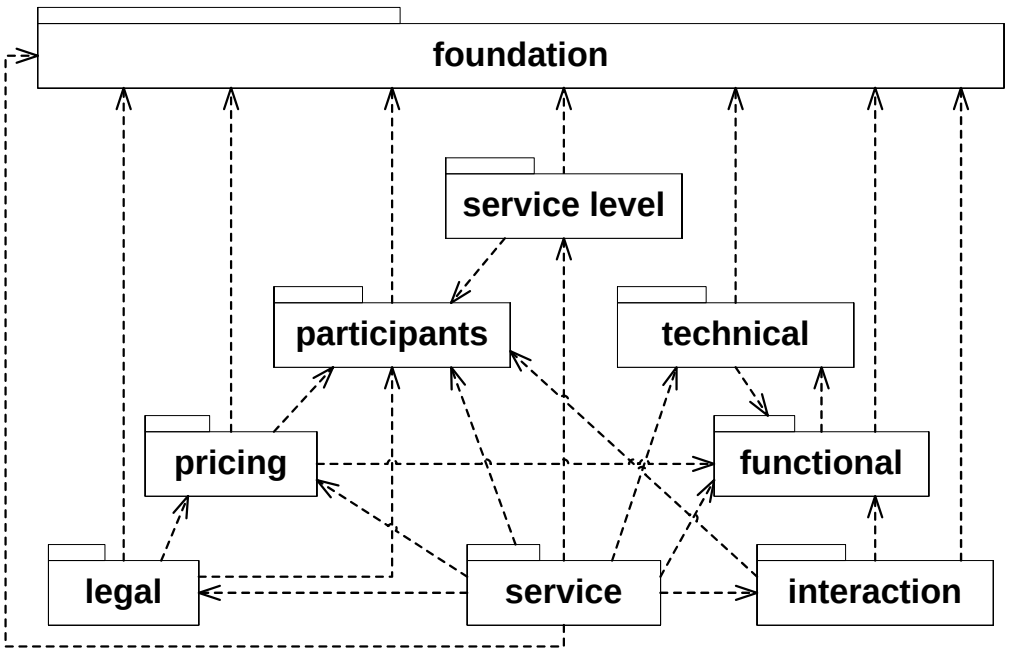

Figure 1 Packages comprising the USDL model and their dependencies (represented as arrows)

About this document

The USDL meta-model is formally defined in Ecore (the meta-modeling language of EMF), with each USDL module being captured in a separate package. This document is one in a series of USDL documents and covers the Service Level Module defined in package “servicelevel”. The series also includes:

- *USDL Overview*
- *Module-specific documentation of the modules Foundation, Service, Participants, Functional, Technical, Interaction, Pricing, and Legal*

The document only provides insights into the concepts of the Service Level Module. For a complete overview of USDL it is recommended to also consider the other documents of the series.

2 Overview

2.1 Introduction to Service Level Module

With the services sector forming a substantial part of developed economies it is all but surprising that there is never just a single service for a particular purpose. Innovation, operational excellence and competition are all key driving forces that make organizations try something new or try to do improve in order to get their share of a market. For consumers of services this means they are often confronted with more than just a few service candidates, which in many cases are similar in terms of offered functionality, if not identical. This makes it difficult for them to choose, and all they usually want is to simply select the service that best suits their needs. Service providers therefore have to establish other distinguishing factors, beyond the core aspect of service functionality. These factors are the non-functional properties and qualities under which a service operates, e.g., general availability (where and when), exception handling procedures, or policies concerning the protection of privacy information. Thus, when offering, negotiating or contracting services, it is of importance to specify these *service levels*.

Service Level Agreements (SLAs) are a common way to formally specify such functional and non-functional conditions under which services are or are to be delivered. However, SLAs in practice are specified at the top-level interface between a service provider and a service customer only. Customers and providers can use top-level SLAs to monitor whether the actual service delivery complies with the agreed SLA terms. In case of SLA violations, penalties or compensations can be directly derived. The USDL Service Level Module allows modeling such information. The module is derived from our research[4] as part of a multilevel SLA management platform.

2.2 General Module Information

Parameters of the package that captures the module

- Namespace: *http://internet-of-services.com/usdl/modules/servicelevel*
- Name: *servicelevel*

The remainder of this section describes the classes and enumerations that are part of the package. A class diagram of the package is depicted in Figure 2. The diagram shows which associations are compositions and which ones are normal relationships. Associations not shown are assumed to be of type composition by default.

[4] Theilmann, W., Happe, J., Kotsokalis, C., Edmonds, A., Kearney, K. & Lambea, J. (2010) A Reference Architecture for Multi-Level SLA Management. Journal of Internet Engineering

Note: Example fragments are provided for some of the classes. In order to improve readability they are presented in XML-based pseudo syntax. This is NOT the official USDL syntax, which is still under development. However, there currently exists a serialization format that is XMI-based and supported through a USDL editor developed by SAP Research.

2.3 Module Dependencies

In order to understand concepts from referenced USDL modules in detail, it is recommended to read the following documents, which cover other USDL modules:

- Service
- Foundation
- Participants

A quick overview of the concepts used in the Service Level Module is given below. This will avoid extensive jumping between documents.

Name	Type	Module	Description
NetworkProvisionedEntity	Abstract EClass	Service	The central concept of the USDL model that represents all entities provisioned into a service network, e.g. service or service bundle
ServiceLevelElementRef	Interface EClass	Foundation	The super type of all USDL classes that represent concepts to which a service level attribute may apply
Description	EClass	Foundation	A generic concept that provides various information elements to describe USDL objects
Expression	EClass	Foundation	A generic concept to specify expressions in an arbitrary expression language
Artifact	EClass	Foundation	A generic concept to include links to USDL-external service metadata, as well as arbitrary documents, files, web pages, etc.
Classification	EClass	Foundation	A generic concept that can be used to classify USDL objects into defined classification systems
TypeReference	EClass	Foundation	A specific type of classification that represents a class/type in a type system, e.g. XML schema instance
VariableDeclaration	EClass	Foundation	A concept to capture declaration of generic variables
Role	EClass	Participant	Serves as the super type of roles that participate in the provisioning and delivery of a network provisioned entity (service or service bundle)

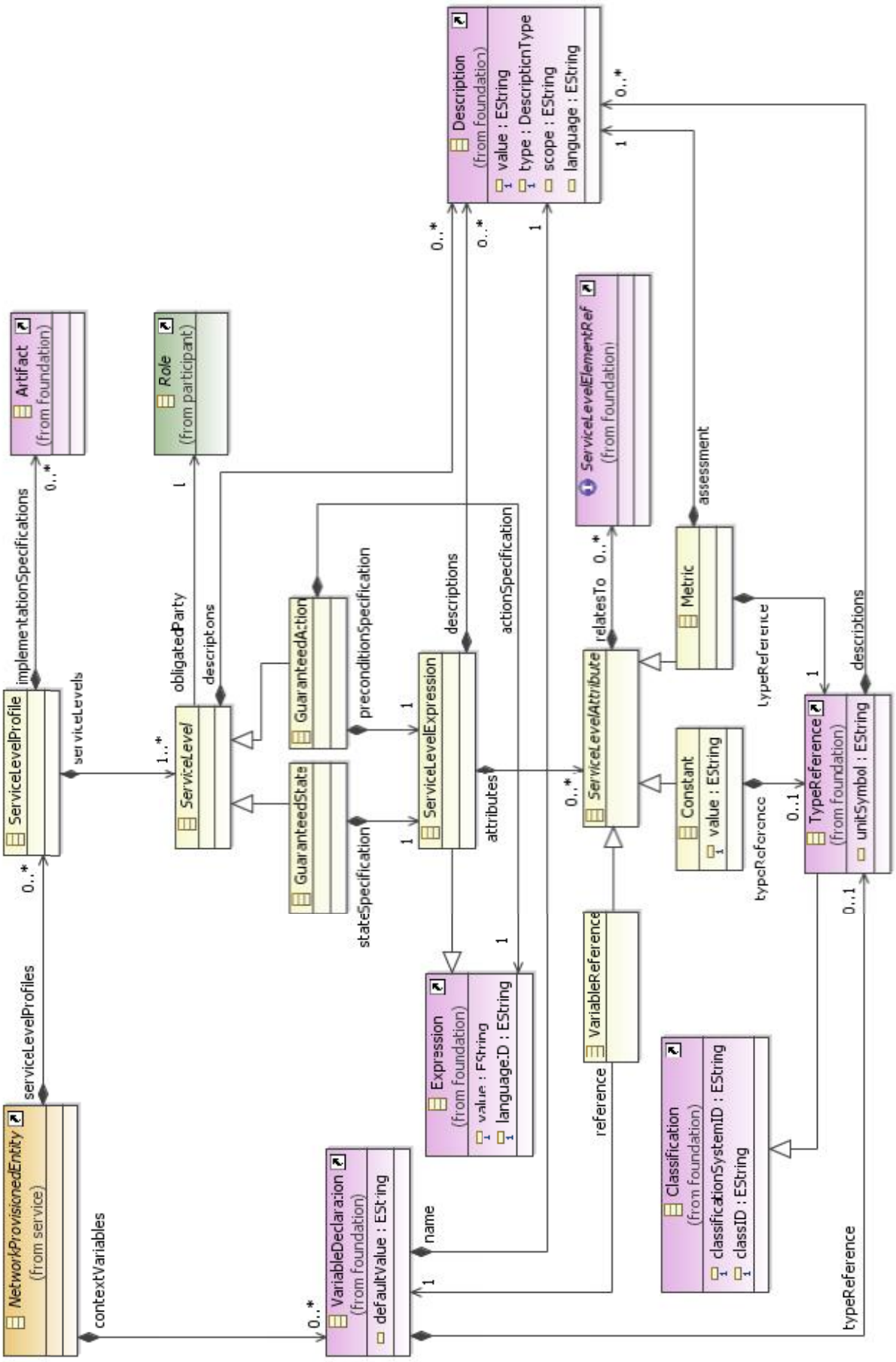

Figure 2 Class diagram of the package that captures the Service Level Module

3 Service Level Module: Model

3.1 ServiceLevelProfile

ServiceLevelProfile represents a set of service level specifications that are combined into one profile and which are offered/negotiated/agreed as a whole. Different profiles can be used to specify different options of how service levels may be specified and grouped (e.g., as gold, silver, bronze profile). A ServiceLevelProfile resembles the concept of a Service Level Agreement Template as for example specified in WS-Agreement.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: N/A

ServiceLevelProfile			
Relations			
Name	Type	Cardinality	Description
serviceLevels	ServiceLevel	1..*	The set of service level specifications grouped in this profile
implementation Specifications	Artifact	0..*	Link to the file(s)/resource(s) that contain either the complete formal definition/specification of this service level profile, or additional metadata that supplements the definitions in USDL

3.2 ServiceLevel

A ServiceLevel specifies a single service level objective as it characterizes an offered, negotiated or agreed service. Service levels are defined by the parties participating in service provisioning, delivery and consumption and express assertions that are claimed or expected to hold during these activities. Such assertions are always attributed to a single party, which is obligated to enforce the service level. From the viewpoint of the party defining the service level two cases are distinguished. Either the defining party obligates itself to ensure the service level, i.e., it claims that the assertion will hold, or the defining party expects the obligated party to ensure the service level, i.e., it requires the other party to enforce the assertion.

- Ecore Type: Abstract EClass
- Interfaces: N/A
- Superclass: N/A

ServiceLevel			
Relations			
Name	Type	Cardinality	Description
obligatedParty	Role	1	The party that is in charge to guarantee/enforce this service level.
descriptions	Description	0..*	Set of (additional) descriptive information about the service level, possibly in multiple natural languages

3.3 GuaranteedState

A GuaranteedState is a particular ServiceLevel that specifies a single state that must be maintained within the lifetime of any service instance, to which the respective service level profile applies.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: ServiceLevel

GuaranteedState			
Relations			
Name	Type	Cardinality	Description
stateSpecification	ServiceLevel Expression	1	An expression that formally specifies the state to be guaranteed

Examples (in pseudo concrete syntax)

```
<identifiableElement xsi:type="service:Service">
...
 <serviceLevelProfiles>
  <serviceLevelProfile>
   <serviceLevels>

    <serviceLevel xsi:type="servicelevel:GuaranteedState">

     <!-- provider is obligated -->
     <obligatedParty> prov321 </obligatedParty>

     <stateSpecification>

      <descriptions>
       <description>
        <value>
          if customer type is "occasional" (5), guarantee average response time of 2 seconds
       </value>
        <type> freetextShort </type>
        <language> en </language>
       </description>
      </descriptions>

      <value>
       if (variable["v_custType"] == constant["c1"]) metric["m_avgRespTime"] < constant["c2"]
      </value>
      <languageID> urn:example:expression_language </languageID>

      <attributes>
       <attribute xsi:id="c1" xsi:type="servicelevel:Constant">
        <value> 5 </value>
       </attribute>
       <attribute xsi:id="c2" xsi:type="servicelevel:Constant">
        <value> 2 </value>
        <typeReference> <!-- duration in seconds --> </typeReference>
       </attribute>
       <attribute xsi:id="m_avgRespTime" xsi:type="servicelevel:Metric">
        <relatesTo> intf8642 </relatesTo>
        <typeReference> <!-- duration in seconds --> </typeReference>
        <assessment> <!—average time between receipt of request and return of response -->
       </assessment>
```

```
        </attribute>

    <attribute xsi:id="v_custType" xsi:type="servicelevel:VariableReference">
          <reference> varCustomerType </reference>
        </attribute>
      </attributes>

    </stateSpecification>

   </serviceLevel>

   <serviceLevels>
  </serviceLevelProfile>
 </serviceLevelProfiles>
...
</identifiableElement>
```

3.4 GuaranteedAction

A GuaranteedAction is a particular ServiceLevel that specifies a self-contained activity that must be performed, if and only if during the lifetime of any service instance to which the respective service level profile applies a specific precondition is fulfilled.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: ServiceLevel

GuaranteedAction			
Relations			
Name	Type	Cardinality	Description
preconditionSpecification	ServiceLevel Expression	1	An expression that formally specifies the precondition to be evaluated
actionSpecification	Expression	1	A formal specification of the action that is taken if the precondition is fulfilled

Examples (in pseudo concrete syntax)

```
<identifiableElement xsi:type="service:Service">
...
 <serviceLevelProfiles>
  <serviceLevelProfile>
   <serviceLevels>

    <serviceLevel xsi:type="servicelevel:GuaranteedAction">
      ...
     <descriptions>
      <description>
       <value>
          if customer type is "premium" (1) and single response time exceeds 1 second, waive service fee
          of this invocation
       </value>
       <type> freetextLong </type>
       <language> en </language>
      </description>
     </descriptions>
     <preconditionSpecification>
```

```
        <!-- customer type is "premium" and response time is longer than 1 second -->
      </preconditionSpecification>

      <actionSpecification>
       <value> <!-- credit the customer with service invocation fee --> </value>
       <languageID> urn:example:action_language </languageID>
      </actionSpecification>

     </serviceLevel>
    <serviceLevels>
   </serviceLevelProfile>
  </serviceLevelProfiles>
...
</identifiableElement>
```

3.5 ServiceLevelExpression

A ServiceLevelExpression specifies an expression that is evaluated in the context of a service level state or action. For this purpose it may reference a set of service level attributes (constants, metrics or variable references) and define relationships between these attributes, e.g. Boolean or arithmetic operands. Typically, it resolves to a Boolean value that indicates whether a guaranteed state is met or whether the precondition to a guaranteed action is fulfilled.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: Expression

ServiceLevelExpression			
Relations			
Name	Type	Cardinality	Description
attributes	ServiceLevel Attribute	0..*	A set of attributes that are used in the formal expression (specified in the attributes inherited from the super class)
descriptions	Description	0..*	Set of (additional) descriptive information about the service level expression, possibly in multiple natural languages

Examples (in pseudo concrete syntax)

```
<serviceLevel xsi:type="servicelevel:GuaranteedAction">
...
 <preconditionSpecification>

  <descriptions>
   <description>
    <value> customer type is "premium" (1) and response time is longer than 1 second </value>
    <type> freetextShort </type>
    <language> en </language>
   </description>
  </descriptions>

   <value> variable["v_custType2"] == constant["c3"] AND metric["m_respTime"] > constant["c4"] </value>
  <languageID> urn:example:expression_language </languageID>

  <attributes>
```

```
    <attribute xsi:id="c3" xsi:type="servicelevel:Constant">
      <value> 1 </value>
     </attribute>
    <attribute xsi:id="c4" xsi:type="servicelevel:Constant">
      <value> 1 </value>
      <typeReference> <!-- duration in seconds --> </typeReference>
    </attribute>
    <attribute xsi:id="m_respTime" xsi:type="servicelevel:Metric">
      <relatesTo> intf8642 </relatesTo>
      <typeReference> <!-- duration in seconds --> </typeReference>
      <assessment> <!—time between receipt of request and return of response --> </assessment>
    </attribute>
    <attribute xsi:id="v_custType2" xsi:type="servicelevel:VariableReference">
      <reference> varCustomerType </reference>
    </attribute>
   </attributes>
 </preconditionSpecification>
...
</serviceLevel>
```

3.6 ServiceLevelAttribute

A ServiceLevelAttribute specifies a single attribute that is part of a service level expression. Attributes can take various concrete forms, of which three (constant, variable reference and metric) are defined in the core version of USDL. A service level attribute has a scope, i.e. it exits in reference to something to which it applies. By default, all attributes are defined in relation to the entire service (including its overall context). Alternatively, a scope that covers only parts thereof may be specified.

- Ecore Type: Abstract EClass
- Interfaces: N/A
- Superclass: N/A

ServiceLevelAttribute			
Relations			
Name	Type	Cardinality	Description
relatesTo	ServiceLevel ElementRef	0..*	Reference to a set of other USDL objects to which the attribute applies or refers to.

3.7 Constant

A Constant specifies a single service level attribute which is constant during service operation, i.e., during the lifetime of any service instance.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: ServiceLevelAttribute

Constant			
Attributes			
Name	Type	Cardinality	Description
value	EString	1	The actual value of the constant
Relations			
Name	Type	Cardinality	Description
typeReference	Type Reference	0..1	A pointer to a an entity in a type schema that formally specifies the structure of the constant

Examples (in pseudo concrete syntax)

```
<serviceLevel xsi:type="servicelevel:GuaranteedState">
 ...
 <stateSpecification>
  ...
  <attributes>
   <attribute xsi:id="c1" xsi:type="servicelevel:Constant">
    <value> 5 </value>
   </attribute>
   <attribute xsi:id="c2" xsi:type="servicelevel:Constant">
    <value> 2 </value>
    <typeReference>
     <classificationSystemID> http://www.moonbank.com/banking/serviceTypes </classificationSystemID>
     <classID> duration_in_seconds </classID>
     <unitSymbol> s </unitSymbol>
     <descriptions>
      <description>
       <value> duration in seconds </value>
       <type> freetextShort </type>
       <language> en </language>
      </description>
     </descriptions>
    </typeReference>
   </attribute>
   ...
  </attributes>
  ...
 </stateSpecification>
</serviceLevel>
```

3.8 Metric

A Metric specifies a single service level attribute which refers to the observation (measure) of a property of the service at service runtime. It may change over the lifetime of a service instance.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: ServiceLevelAttribute

Metric			
Attributes			
Name	Type	Cardinality	Description
assessment	Description	0..1	Descriptive information about how the metric is assessed/measured; *constraint:* description type has to be set to *freetextLong*

Relations			
Name	Type	Cardinality	Description
typeReference	Type Reference	0..1	A pointer to a an entity in a type schema that formally specifies the structure of the metric

Examples (in pseudo concrete syntax)

```
<serviceLevel xsi:type="servicelevel:GuaranteedState">
 ...
 <stateSpecification>
  ...
  <attributes>
   <attribute xsi:id="m_avgRespTime" xsi:type="servicelevel:Metric">
    <relatesTo> intf8642 </relatesTo>
    <typeReference> <!-- duration in seconds --> </typeReference>
    <assessment>
     <value> response time of service in seconds measured from reception of request at gateway server
     until response completely returned, averaged over 1 hour intervals</value>
     <type> freetextLong </type>
     <language> en </language>
    </assessment>
   </attribute>
   ...
  </attributes>
  ...
 </stateSpecification>
</serviceLevel>
```

3.9 VariableReference

A VariableReference allows for referencing a variable declared in the global context of a service or service bundle. Variable references are used, for instance, as part of service level expressions.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: ServiceLevelAttribute

VariableReference			
Relations			
Name	Type	Cardinality	Description
reference	Variable Declaration	1	Reference to the declaration of the referenced variable

Examples (in pseudo concrete syntax)

```
<serviceLevel xsi:type="servicelevel:GuaranteedState">
 ...
 <stateSpecification>
  ...
  <attributes>
   <attribute xsi:id="v_custType" xsi:type="servicelevel:VariableReference">
    <reference> varCustomerType </reference>
   </attribute>
  </attributes>
 </stateSpecification>
</serviceLevel>
```

4 Basic Extension of the Service Level Module

As outlined in the overview section of this specification, it was decided to take a generic approach for the definition of the Service Level Module and then supplement this part of the USDL language with a basic extension that introduces a set of concretely specified service level attributes, in particular metrics. With the Service Level Module specified in Section 3, this set can now be documented.

It should be pointed out again that this extension is not part of the core USDL language. It is a proper extension, which by default is included in the language specification of USDL. No implementation of the USDL specification is required to support this extension.

Parameters of the package that captures the extension

- Namespace: *http://internet-of-services.com/usdl/extensions/servicelevelbaseextension*
- Name: *servicelevelbaseextension*

The remainder of this section describes the classes and enumerations that are part of the package. A class diagram of the package is depicted in Figure 3. The diagram shows which associations are compositions and which ones are normal relationships. Associations not shown are assumed to be of type composition by default.

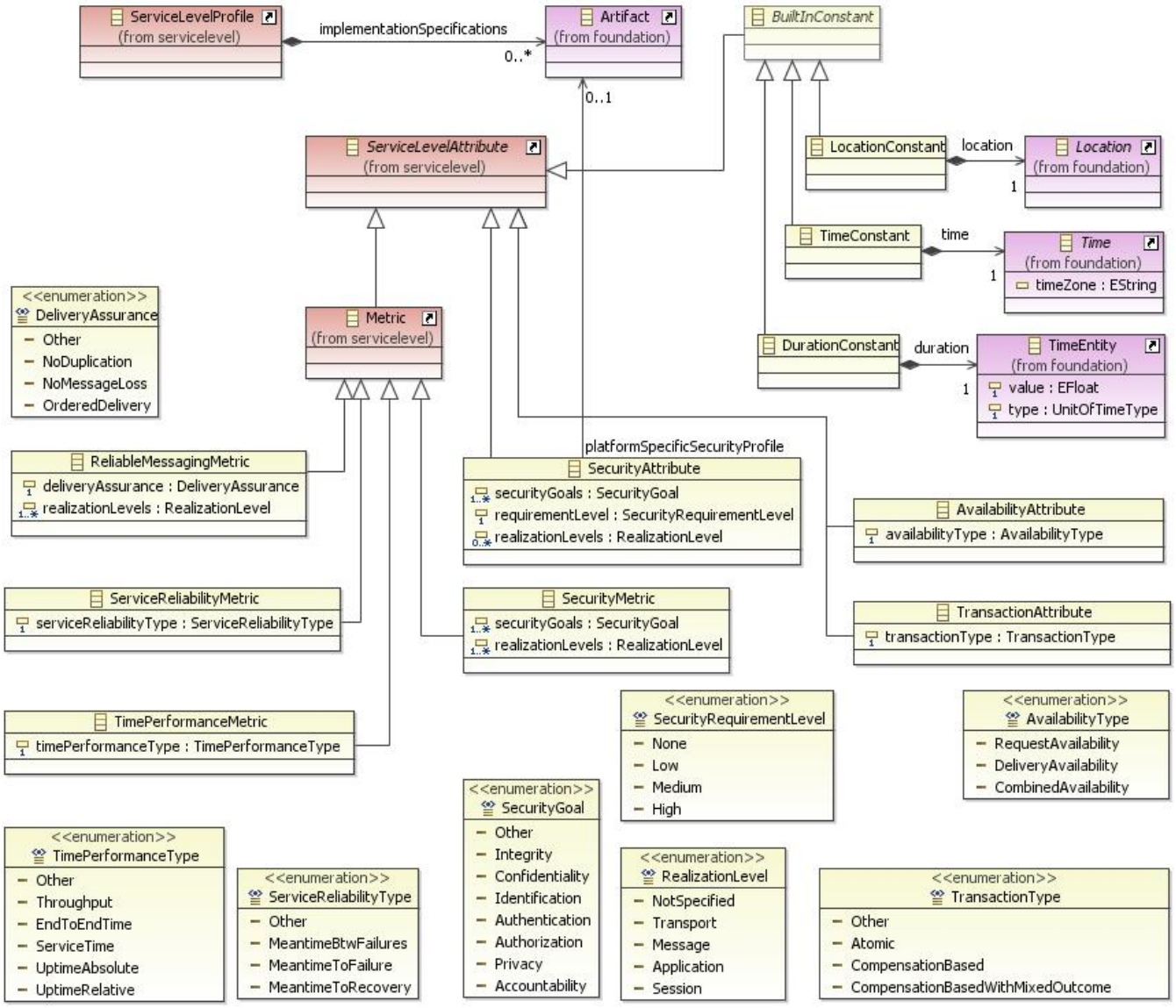

Figure 3 Class diagram of the package that captures the basic extension of the Service Level Module

4.1 BuiltInConstant

BuiltInConstant is a special type of constant that should be used for constant values that can be expressed using classes from the USDL foundation, e.g., time constants, location constants and duration constants. On a conceptual level this class is the same as Constant. BuiltInConstant is introduced as a parallel concept in order to capitalize on the investments made in the foundation of USDL.

- Ecore Type: Abstract EClass
- Interfaces: N/A
- Superclass: ServiceLevelAttribute

4.2 LocationConstant

LocationConstant is a specific type of BuiltInConstant. It references a location object, which represents the value of the constant.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: BuiltInConstant

LocationConstant			
Relations			
Name	Type	Cardinality	Description
location	Location	1	The value of the constant (a location object)

Examples (in pseudo concrete syntax)

```
<serviceLevel xsi:type="servicelevel:GuaranteedState">
 ...
 <stateSpecification>
  ...
  <attributes>
   <attribute xsi:id="c_loc1" xsi:type="slbaseext:LocationConstant">
    <location xsi:type="foundation:AdministrativeArea">
     <value> US </value>
     <type> country </type>
     <descriptions>
      <description>
       <value> United States of America </value>
       <type> name </type>
       <language> en </language>
      </description>
     </descriptions>
    </location>
   </attribute>
   ...
  </attributes>
  ...
 </stateSpecification>
</serviceLevel>
```

4.3 TimeConstant

TimeConstant is a specific type of BuiltInConstant. It references a time object, which represents the value of the constant.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: BuiltInConstant

TimeConstant			
Relations			
Name	Type	Cardinality	Description
time	Time	1	The value of the constant (a time object)

Examples (in pseudo concrete syntax)

```
<serviceLevel xsi:type="servicelevel:GuaranteedState">
 ...
 <stateSpecification>
  ...
  <attributes>
   <attribute xsi:id="c_time1" xsi:type="slbaseext:TimeConstant">
    <time xsi:type="foundation:DurationInterval">

     <start xsi:type="foundation:AbsolutePointInTime">
      <value> 2010-01-01T02:00:00.000-05:00 </value>
     </start>

     <intervalDuration>
      <type> year </type>
      <value> 3.0 </value>
     </intervalDuration>

     <descriptions>
      <description>
       <value> Service Validity Period </value>
       <type> name </type>
       <language> en </language>
      </description>
      <description>
       <value> Interval that starts on Jan. 1, 2010, 2:00am EST, and lasts for 3 years </value>
       <type> freetextShort </type>
       <language> en </language>
      </description>
     </descriptions>

    </time>
   </attribute>
   ...
  </attributes>
  ...
 </stateSpecification>
</serviceLevel>
```

4.4 DurationConstant

DurationConstant is a specific type of BuiltInConstant. It references a time entity object, which represents the value of the constant (a duration).

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: BuiltInConstant

DurationConstant			
Relations			
Name	Type	Cardinality	Description
duration	TimeEntity	1	The value of the constant (a duration object)
Examples (in pseudo concrete syntax)			

```
<serviceLevel xsi:type="servicelevel:GuaranteedState">
 ...
 <stateSpecification>
  ...
  <attributes>
   <attribute xsi:id="c_dur1" xsi:type="slbaseext:DurationConstant">
    <duration>
     <type> second </type>
     <value> 2.0 </value>
    </duration>
   </attribute>
   ...
  </attributes>
  ...
 </stateSpecification>
</serviceLevel>
```

4.5 RealizationLevel

RealizationLevel is used to express at which layer in the service's implementation stack a service level attribute is realized, e.g. at which layer are the measurements of a metric taken.

- Ecore Type: EEnum

RealizationLevel		
Items		
Name	Int Value	Description
NotSpecified	0	Realization level is not specified (unknown)
Transport	1	Attribute is realized on the transport layer in the implementation stack
Message	2	Attribute is realized on the message layer in the implementation stack
Application	3	Attribute is realized by the application layer in the implementation stack, e.g. application (framework) implementing the service
Session	4	Attribute is realized by the session layer in the implementation stack, e.g. session management component in the application framework implementing the service

4.6 ServiceReliabilityMetric

ServiceReliabilityMetric is a specific type of Metric. It describes various metrics that capture quantifiable guarantees of service reliability, i.e. to which extent may a consumer assume that either the service or those parts of it that are in the scope of this service level attribute (cf. Section 3.6) is/are properly functioning at any given point in time.

Note: The *type* attribute inherited from Metric has to be compatible with DurationConstant objects.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: Metric

ServiceReliabilityMetric			
Attributes			
Name	Type	Cardinality	Description
serviceReliabilityType	Service ReliabilityType	1	Indicates the specific service reliability metric expressed by this attribute

Examples (in pseudo concrete syntax)

```
<serviceLevel xsi:type="servicelevel:GuaranteedState">
 ...
 <stateSpecification>
  ...
  <value> metric["m_servRel1"] >= constant["c11"] </value>
  ...
  <attributes>
   <attribute xsi:id="m_servRel1" xsi:type="slbaseext:ServiceReliabilityMetric">
    <serviceReliabilityType> MeantimeBtwFailures </serviceReliabilityType>
    <typeReference> <!-- duration in days --> </typeReference>
   </attribute>
   <attribute xsi:id="c11" xsi:type="slbaseext:DurationConstant">
    <duration>
     <type> day </type>
     <value> 180 </value>
    </duration>
   </attribute>
   ...
  </attributes>
  ...
 </stateSpecification>
</serviceLevel>
```

4.7 ServiceReliabilityType

ServiceReliabilityType determines which service reliability metric is expressed by an instance of ServiceReliabilityMetric.

- Ecore Type: EEnum

ServiceReliabilityType		
Items		
Name	Int Value	Description
Other	0	Service reliability type is not listed or not known

MeantimeBtw Failures	1	This service reliability type is defined as the average time between two occurrences of failure of a service instance (deployed service) to execute correctly – with the service instance being repaired instantly after a failure
MeantimeToFailure	2	This service reliability type is defined as the average time for a service instance (deployed service) to fail – with the instance being replaced (re-deployed) after failure
MeantimeTo Recovery	3	This service reliability type is defined as the average time for a service instance (deployed service) to be repaired

4.8 TimePerformanceMetric

TimePerformanceMetric is a specific type of Metric. It describes various metrics that capture quantifiable guarantees of performance, which are related to the timeliness of rendering the service and the service's accessibility within specified availability ranges. These guarantees apply to those parts of the service that are in the scope of this service level attribute (cf. Section 3.6)

Note: The *type* attribute inherited from Metric has to be compatible with a type that is implied by the specific metric chosen when instantiating an object of this class.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: Metric

TimePerformanceMetric			
Attributes			
Name	Type	Cardinality	Description
timePerformanceType	Time Performance Type	1	Indicates the specific time performance metric expressed by this attribute

Examples (in pseudo concrete syntax)

```
<serviceLevel xsi:type="servicelevel:GuaranteedState">
 ...
 <stateSpecification>
  ...
  <value> metric["m_timePerf1"] <= constant["c12"] </value>
  ...
  <attributes>
   <attribute xsi:id="m_timePerf1" xsi:type="slbaseext:TimePerformanceMetric">
    <timePerformanceType> EndToEndTime </timePerformanceType>
    <typeReference> <!-- duration in milliseconds --> </typeReference>
   </attribute>
   <attribute xsi:id="c12" xsi:type="slbaseext:DurationConstant">
    <duration>
     <type> millisecond </type>
     <value> 800 </value>
    </duration>
   </attribute>
   ...
  </attributes>
  ...
 </stateSpecification>
</serviceLevel>
```

4.9 TimePerformanceType

TimePerformanceType determines which time-related performance metric is expressed by an instance of TimePerformanceMetric.

- Ecore Type: EEnum

TimePerformanceType		
Items		
Name	Int Value	Description
Other	0	Time-related performance type is not listed or not known
Throughput	1	This time-related performance type is defined as the number of successful service executions in a given time; *constraint:* metric type has to be compatible with types in the form of number (of requests/executions) per duration (objects of class DurationConstant)
EndToEndTime	2	This time-related performance type is defined as the time it takes between the receipt of a service request and completely returning a response (performing the service in-between); *constraint:* metric type has to be compatible with objects of class DurationConstant
ServiceTime	3	This time-related performance type is defined as the time it takes to just perform a service; *constraint:* metric type has to be compatible with objects of class DurationConstant
UptimeAbsolute	4	This time-related performance type is defined as the (absolute) time a service is continuously accessible in its given availability limits; *constraint:* metric type has to be compatible with objects of class DurationConstant
UptimeRelative	5	This time-related performance type is defined as the percentage of time a service is accessible in its given availability limits; *constraint:* metric type has to be compatible with percentage-based types

4.10 SecurityMetric

SecurityMetric is a specific type of Metric. It is used to capture guarantees about specific security measures put in place by the parties involved in provisioning, delivery and consumption of a service, e.g. authentication methods or encryption schemes. Measures are applied to those parts of the service that are in the scope of this service level attribute (cf. Section 3.6). As measures can usually be attributed to one or more specific security goals, it is mandatory that objects of this class define at least one such goal.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: Metric

SecurityMetric			
Attributes			
Name	Type	Cardinality	Description
securityGoals	SecurityGoal	1..*	Indicates which set of security goals are addressed by this metric
realizationLevels	Realization Level	1..*	Indicates the layer in the service's implementation stack at which the security metric is realized

Examples (in pseudo concrete syntax)

```
<serviceLevel xsi:type="servicelevel:GuaranteedState">
 ...
 <stateSpecification>
  ...
  <value> metric["m_sec1"] == constant["c13"] OR metric["m_sec1"] == constant["c14"] </value>
  ...
  <attributes>

   <attribute xsi:id="m_sec1" xsi:type="slbaseext:SecurityMetric">
    <relatesTo> intf8642 </relatesTo>
    <securityGoals>
     <securityGoal> Authentication </securityGoal>
    <securityGoals>
    <realizationLevels>
     <realizationLevel> Session </realizationLevel>
    <realizationLevels>
    <typeReference> <!-- untyped constant --> </typeReference>
   </attribute>

   <attribute xsi:id="c13" xsi:type="servicelevel:Constant">
    <value> HTTP BASIC </value>
   </attribute>
   <attribute xsi:id="c14" xsi:type="servicelevel:Constant">
    <value> HTTP DIGEST </value>
   </attribute>
  </attributes>
  ...
 </stateSpecification>
</serviceLevel>
```

4.11 SecurityAttribute

SecurityAttribute is a service level attribute that describes guarantees about security in an abstract way. Thus, unlike SecurityMetric, it does not provide reference to specific technical security measures, but specifies requirements on service security in terms of security goals and security requirement levels. These requirements are then interpreted in the context of the particular platform that provides access to the service, using a platform-specific security profile (additional service metadata).

Providing two dedicated ways for the specification of security guarantees offers greater flexibility to different user groups of USDL. Using SecurityAttribute to associate security goals with requirement levels is an alternative to the formal specification of concrete security measures via SecurityMetric. This alternative is thought as a simple and non-technical way to express business service security characteristics. It enables actors (on the provider as well as on the consumer side) who are not "technical security experts", i.e. who are not familiar with e.g. WS-Security policies or XACML statements, to express their security demands.

A typical use case for the specification of business service security characteristics by means of security goals and corresponding security requirement levels is the provision and consumption of business services on a platform that is operated by a third party.

Such a platform, e.g. a service marketplace, as part of its infrastructure usually provides shared security services and identity management services to business service providers and consumers. The platform may undertake the task of implementing the security goals at the desired requirement levels by generating appropriate technical security policies (e.g. WS-Securitypolicy artifacts) and by

providing suitable infrastructure services that are handling authentication, authorization, encryption etc.

In this case the platform operator may create a platform-specific security profile to specify the technical details on how a security goal of a given security requirement level is realized. The platform-specific security profile maps security goals and security requirement levels to concrete security mechanisms and technical standards that are supported by the platform. For example, a platform operator may map the security goal Authentication with the security requirement level low to a formal WS-SecurityPolicy statement which specifies that a UsernameToken with a password digest and a creation timestamp is required to be authenticated. The security goal Authentication with the security requirement level high would e.g. be mapped to a WS-SecurityPolicy statement demanding an X.509-Token.

The security mechanisms as well as the mapping of security goals and requirement levels to security mechanisms and technical standards are likely to vary from platform to platform. For this reason, USDL does not prescribe the form or structure of a platform-specific security profile. It is simply referenced.

In summary, the specification of security requirement levels for security goals enables service providers to express business service security characteristics on a conceptual rather than on a technical level. The same applies to service consumers that may search for appropriate business services based on these abstract characteristics.

Specific measures determined from the platform-specific security profile are applied to those parts of the service that are in the scope of this service level attribute (cf. Section 3.6).

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: ServiceLevelAttribute

SecurityAttribute			
Attributes			
Name	Type	Cardinality	Description
securityGoals	SecurityGoal	1..*	Indicates which set of security goals are to be addressed by the measures satisfying the requirement set in this security attribute
realizationLevels	Realization Level	0..*	Indicates at which layer in the service's implementation stack concrete security measures should be implemented
requirementLevel	Security Requirement Level	1	Indicates the required implementation degree (i.e. the level of security/protection) with respect to the security goals addressed
Relations			
Name	Type	Cardinality	Description
platformSpecificSecurity Profile	Artifact	0..1	An (optional) platform specific security profile defining platform specific security mechanisms, security metrics and classifications; *constraint:* has to point to one of the artifacts listed as implementation specifications of the service level profile

Examples (in pseudo concrete syntax)

```
<serviceLevelProfile>

 <implementationSpecifications>
  <implementationSpecification xsi:id=”profile123”>
   <type> TechnicalMetadata </type>
   <mimeType> application/xml </mimeType>


   <uri> http://serviceontology.org/services/security/profile123 </uri>

    <descriptions>
     <description>
      <value> security profile 123 </value>
      <type> name </type>
      <language> en </language>
     </description>
    </descriptions>
  </implementationSpecification>
 </implementationSpecifications>
 ...
 <serviceLevels>
  <serviceLevel xsi:type=”servicelevel:GuaranteedState”>
    ...
    <stateSpecification>
     ...
     <value> attribute[“sec2”] applies </value>
     <value> attribute[“sec3”] applies </value>
     ...
     <attributes>

      <attribute xsi:id=”sec2” xsi:type=”slbaseext:SecurityAttribute”>
       <relatesTo> intf8642 </relatesTo>
       <securityGoals>
        <securityGoal> Authentication </securityGoal>
       <securityGoals>
       <requirementLevel> low </requirementLevel>
       <platformSpecificSecurityProfile> profile123 </platformSpecificSecurityProfile>
      </attribute>
      <attribute xsi:id=”sec3” xsi:type=”slbaseext:SecurityAttribute”>
       <relatesTo> intf8642 </relatesTo>
       <securityGoals>
        <securityGoal> Confidentiality </securityGoal>
       <securityGoals>
       <requirementLevel> medium </requirementLevel>
       <platformSpecificSecurityProfile> profile123 </platformSpecificSecurityProfile>
      </attribute>
     </attributes>
     ...
    </stateSpecification>
  </serviceLevel>
 <serviceLevels>
<serviceLevelProfile>
```

4.12 SecurityGoal

SecurityGoal is used to associate a SecurityMetric or SecurityAttribute with the set of security goals that are addressed.

- Ecore Type: EEnum

SecurityGoal		
Items		
Name	Int Value	Description
Other	0	The addressed security goal is not listed or not known
Integrity	1	The addressed security goal is integrity, i.e. safeguarding a service against unauthorized manipulation
Confidentiality	2	The addressed security goal is confidentiality, i.e. safeguarding a service against unauthorized access to information (eavesdropping)
Identification	3	The addressed security goal is identification, i.e. ensuring that the identity of a subject has been verified (e.g. as part of a registration procedure)
Authentication	4	The addressed security goal is authentication, i.e. ensuring that only authenticated subjects gain access to a service
Authorization	5	The addressed security goal is authorization, i.e. ensuring that identified subjects gain access to only those parts of a service they are authorized for
Privacy	6	The addressed security goal is privacy, i.e. ensuring that personal data available in a service's context is only proliferated to authorized parties
Accountability	7	The addressed security goal is accountability, i.e. ensuring that activity in a service's context can be safely identified and attributed to a subject

4.13 SecurityRequirementLevel

SecurityRequirementLevel is used to indicate the implementation degree, i.e. the level of protection/security with respect to the security goal(s) addressed by a security attribute.

- Ecore Type: EEnum

SecurityRequirementLevel		
Items		
Name	Int Value	Description
None	0	The implementation degree of a given security goal is not listed or not known
Low	1	The implementation degree of a given security goal is low
Medium	2	The implementation degree of a given security goal is medium
High	3	The implementation degree of a given security goal is high

4.14 ReliableMessagingMetric

ReliableMessagingMetric is a metric that describes the quality of assured message delivery between consumer and service. Reliable messaging features are applied to those parts of the service that are in the scope of this metric (cf. Section 3.8).

Note: This metric is not quantifiable. It either applies to its assigned scope or not.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: Metric

ReliableMessagingMetric			
Attributes			
Name	Type	Cardinality	Description
deliveryAssurance	Delivery Assurance	1	Indicates the level of assurance applied for delivering messages to a recipient
realizationLevels	Realization Level	1..*	Indicates the layer in the service's implementation stack at which reliable messaging is realized

Examples (in pseudo concrete syntax)

```
<serviceLevel xsi:type="servicelevel:GuaranteedState">
 ...
 <stateSpecification>
  ...
  <value> attribute["relMsg1"] applies </value>
  ...
  <attributes>
   <attribute xsi:id="relMsg1" xsi:type="slbaseext:ReliableMessagingAttribute">
    <relatesTo> intf8642 </relatesTo>
    <deliveryAssurance> NoDuplication </deliveryAssurance>
    <realizationLevels>
     <realizationLevel> Message </realizationLevel>
    <realizationLevels>
   </attribute>
   ...
  </attributes>
  ...
 </stateSpecification>
</serviceLevel>
```

4.15 DeliveryAssurance

DeliveryAssurance indicates the level of assurance which is applied to reliable messaging.

- Ecore Type: EEnum

DeliveryAssurance		
Items		
Name	Int Value	Description
Other	0	Level of delivery assurance is not listed or not known
NoDuplication	1	Messages are delivered in a way which ensures that no duplicates occur
NoMessageLoss	2	Messages are delivered in a way which ensures that no message is lost
OrderedDelivery	3	Messages are delivered in a way which ensures the order of their sending

4.16 TransactionAttribute

TransactionAttribute is a service level attribute that describes transactional features supported in the context of performing a service. Transactional behavior applies to those parts of the service that are in the scope of this service level attribute (cf. Section 3.6).

Note: This attribute is not quantifiable. It either applies to its assigned scope or not.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: ServiceLevelAttribute

TransactionAttribute			
Attributes			
Name	Type	Cardinality	Description
transactionType	Transaction Type	1	Indicates a particular transactional behavior applied to service execution
Examples (in pseudo concrete syntax)			

```
<serviceLevel xsi:type="servicelevel:GuaranteedState">
 ...
 <stateSpecification>
  ...
  <value> attribute["trans1"] applies </value>
  ...
  <attributes>
   <attribute xsi:id="trans1" xsi:type="slbaseext:TransactionAttribute">
    <transactionType> CompensationBased </transactionType>
   </attribute>
   ...
  </attributes>
  ...
 </stateSpecification>
</serviceLevel>
```

4.17 TransactionType

TransactionType indicates the specific transactional behavior that is applied when performing a service (or parts of a service).

- Ecore Type: EEnum

TransactionType		
Items		
Name	Int Value	Description
Other	0	Particular transactional behavior is not listed or not known
Atomic	1	A service (or parts thereof) is either performed completely and successfully, or, in case of a failure, it is not regarded as executed at all
CompensationBased	2	In case of a failure/exception during execution of a service (or parts thereof), appropriate actions are taken to undo any effects caused during execution, in order to completely restore the original state
CompensationBased WithMixedOutcome	3	In case of a failure/exception during execution of a service (or parts thereof), appropriate actions are taken to undo any effects caused during execution, in order to restore the original state as much as possible

4.18 AvailabilityAttribute

AvailabilityAttribute is a service level attribute that captures aspects of temporal and geographic availability, i.e. where and when a service is supposedly available. This includes possible times and locations for request and/or delivery. Availability applies to those parts of the service that are in the scope of this service level attribute (cf. Section 3.6).

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: ServiceLevelAttribute

AvailabilityAttribute			
Attributes			
Name	Type	Cardinality	Description
availabilityType	Availability Type	1	Determines whether this attribute refers to availability for request, for delivery, or both.

Examples (in pseudo concrete syntax)

```
<serviceLevel xsi:type="servicelevel:GuaranteedState">
 ...
 <stateSpecification>

  <descriptions>
   <description>
    <value> The service will be available in the US from Jan. 1, 2010, 2:00am EST, for 3 years </value>
    <type> freetextShort </type>
    <language> en </language>
   </description>
  </descriptions>

  <value> attribute["avail1"] == (attribute["c_loc1"] + attribute["c_time1"]) </value>
  <languageID> urn:example:expression_language </languageID>

  <attributes>

   <attribute xsi:id="avail1" xsi:type="slbaseext:AvailabilityAttribute">
    <availabilityType> CombinedAvailability </availabilityType>
   </attribute>

   <attribute xsi:id="c_loc1" xsi:type="slbaseext:LocationConstant">
    <location xsi:type="foundation:AdministrativeArea">
     <value> US </value>
     <type> country </type>
     <descriptions>
      <description>
       <value> United States of America </value>
       <type> name </type>
       <language> en </language>
      </description>
     </descriptions>
    </location>
   </attribute>

   <attribute xsi:id="c_time1" xsi:type="slbaseext:TimeConstant">
    <time xsi:type="foundation:DurationInterval">
```

```
        <start xsi:type="foundation:AbsolutePointInTime">
          <value> 2010-01-01T02:00:00.000-06:00 </value>
        </start>
        <intervalDuration>
          <type> year </type>
          <value> 3.0 </value>
        </intervalDuration>
        <descriptions>
          <description>
            <value> Service Validity Period </value>
            <type> name </type>
            <language> en </language>
          </description>
        </descriptions>
      </time>
    </attribute>

  </attributes>
  ...
 </stateSpecification>
</serviceLevel>
```

4.19 AvailabilityType

AvailabilityType determines the scope of an availability attribute regarding where a service can be requested from and delivered to, and when either of these two principal procedures can take place. Even though availability usually relates to the whole service, there could be scoped applied, e.g. to restrict availability to certain access mechanisms (used for request or delivery) or certain service functions (during delivery).

- Ecore Type: EEnum

TransactionType		
Items		
Name	Int Value	Description
RequestAvailability	0	The availability attribute refers to where a service (or parts of the service) can be requested from and at what times this is possible
DeliveryAvailability	1	The availability attribute refers to where a service (or parts thereof) can be delivered to and at what times delivery happens, or alternatively (additionally) wherefrom and when access to the service is possible
CombinedAvailability	2	The availability attribute refers to both types of availability

5 Examples

In the following two examples are provided in order to draw a more comprehensive example of how the Service Level Module can be used. The first shows how to model the service levels of a human service in the logistics domain. The second shows an example of an automated service.

5.1 Human service

The following XML snippet shows a service level specification of a logistics service, where

- customers can specify their expected delivery duration
- a provider-obligated term guarantees the delivery within the specified duration (#1)
- a provider-obligated term guarantees that goods are maintained at -5 degrees Celsius (#2)
- a provider-obligated action regulates penalty payments for delayed deliveries (#3)

Examples (in pseudo concrete syntax)

```
<identifiableElement xsi:type="service:Service">
...
 <contextVariables>
  <!-- variable for expected duration of delivery in weeks, default: 1 week -->
  <variableDeclaration xsi:id="varExpDelDuration">
   <name>
    <value> expectedDeliveryDuration </value>
    <type> name </type>
   </name>
   <defaultValue> 1 </defaultValue>
   <typeReference>
    <classificationSystemID> http://www.internet-of-services.com/serviceTypes </classificationSystemID>
    <classID> duration_week </classID>
    <unitSymbol> wk </unitSymbol>
    <descriptions>
     <description>
      <value> duration in weeks </value>
      <type> freetextShort </type>
      <language> en </language>
     </description>
    </descriptions>
   </typeReference>
  </variableDeclaration>
 </contextVariables>

 <serviceLevelProfiles>
  <serviceLevelProfile>

   <serviceLevels>

    <!-- service level #1 -->
    <serviceLevel xsi:type="servicelevel:GuaranteedState">
     <!-- provider is obligated -->
     <obligatedParty> prov321 </obligatedParty>

     <descriptions>
      <description>
       <value> Delivery duration must not longer than what has been specified by the customer. </value>
       <type> freetextShort </type>
```

```
        <language> en </language>
       </description>
      </descriptions>

      <stateSpecification>

       <!-- measured delivery duration is less than or equal to what is set in the service context (cust. input -->
       <value> metric[“m_delDuration”] <= variable[“v_expDelDur”] </value>
       <languageID> urn:example:expression_language </languageID>

       <attributes>
        <!-- reference to the variable -->
        <attribute xsi:id=”v_expDelDur” xsi:type=”servicelevel:VariableReference”>
         <reference> varExpDelDuration </reference>
        </attribute>
        <!-- metric for measuring delivery duration  -->
        <attribute xsi:id=”m_delDuration” xsi:type=”servicelevel:Metric”>
         <typeReference>
          <classificationSystemID>
               http://www.internet-of-services.com/serviceTypes
          </classificationSystemID>
          <classID> duration_day </classID>
          <unitSymbol> d </unitSymbol>
          <descriptions>
           <description>
            <value> duration in days </value>
            <type> freetextShort </type>
            <language> en </language>
           </description>
          </descriptions>
         </typeReference>
         <assessment>
          <value> delivery duration as measured by receiving party </value>
          <type> freetextLong </type>
          <language> en </language>
         </assessment>
        </attribute>
       </attributes>

      <stateSpecification>
     </serviceLevel>

     <!-- service level #2 -->
     <serviceLevel xsi:type=”servicelevel:GuaranteedState”>
      <!-- provider is obligated -->
      <obligatedParty> prov321 </obligatedParty>

      <descriptions>
       <description>
        <value> The temperature of the goods is maintained at –5 degrees. </value>
        <type> freetextShort </type>
        <language> en </language>
       </description>
      </descriptions>

<stateSpecification>

       <!-- measured temperature is approximately equal to -5 degrees Celsius -->
```

```
        <value> metric[“m_temp”] ~= constant[“c5”] </value>
        <languageID> urn:example:expression_language </languageID>

        <attributes>
         <!-- temperature threshold (modeled as constant) -->
         <attribute xsi:id=”c5” xsi:type=”servicelevel:Constant”>
          <value> -5 </value>
          <typeReference>
           <classificationSystemID>
            http://www.internet-of-services.com/serviceTypes
           </classificationSystemID>
           <classID> temperature_celsius </classID>
           <unitSymbol> °C </unitSymbol>
           <descriptions>
            <description>
             <value> temperature in degrees Celsius </value>
             <type> freetextShort </type>
             <language> en </language>
            </description>
           </descriptions>
          </typeReference>
         </attribute>
         <!-- metric for measuring temperature, related to goods (input to service function “Transport”)  -->
         <attribute xsi:id=”m_temp” xsi:type=”servicelevel:Metric”>
          <relatesTo> paramGoods </relatesTo>
          <typeReference>
           <classificationSystemID>
            http://www.internet-of-services.com/serviceTypes
           </classificationSystemID>
           <classID> temperature_celsius </classID>
           <unitSymbol> °C </unitSymbol>
           <descriptions>
            <description>
             <value> temperature in degrees Celsius </value>
             <type> freetextShort </type>
             <language> en </language>
            </description>
           </descriptions>
          </typeReference>
          <assessment>
           <value> temperature of goods as constantly measured by provider </value>
           <type> freetextLong </type>
           <language> en </language>
          </assessment>
         </attribute>
        </attributes>

      <stateSpecification>
     </serviceLevel>

     <!-- service level #3 -->
     <serviceLevel xsi:type=”servicelevel:GuaranteedAction”>
      <!-- provider is obligated -->
      <obligatedParty> prov321 </obligatedParty>

<descriptions>
        <description>
         <value> Penalty payment of €100 per full day of delayed delivery </value>
```

```
          <type> freetextShort </type>
          <language> en </language>
         </description>
        </descriptions>

        <preconditionSpecification>
         <!-- measured delivery delay is greater or equal than 1 -->
         <value> metric[“m_delDelay”] >= constant[“c6”] </value>
         <languageID> urn:example:expression_language </languageID>

         <attributes>
          <!-- delivery delay threshold (modeled as constant) -->
          <attribute xsi:id=”c6” xsi:type=”servicelevel:Constant”>
           <value> 1 </value>
           <typeReference> <!-- same type description as specified in metric below --> </typeReference>
          </attribute>
          <!-- metric for measuring delay in delivery  -->
          <attribute xsi:id=”m_delDelay” xsi:type=”servicelevel:Metric”>
           <typeReference>
            <classificationSystemID>
                http://www.internet-of-services.com/serviceTypes
            </classificationSystemID>
            <classID> duration_day </classID>
            <unitSymbol> d </unitSymbol>
            <descriptions>
             <description>
              <value> duration in days </value>
              <type> freetextShort </type>
              <language> en </language>
             </description>
            </descriptions>
           </typeReference>
           <assessment>
            <value> delayed delivery as measured by receiving party </value>
            <type> freetextLong </type>
            <language> en </language>
           </assessment>
          </attribute>
         </attributes>
        <preconditionSpecification>

        <actionSpecification>
         <value> <!-- credit the customer with (€100 * floor(metric[“m_delDelay”])) --> </value>
         <languageID> urn:example:action_language </languageID>
        </actionSpecification>

       </serviceLevel>
      </serviceLevels>

    </serviceLevelProfile>
   </serviceLevelProfiles>
...
</identifiableElement>
```

5.2 Automated Service

The following XML snippet depicts a service level description for an automated service where

- customers can specify the preferred availability and their expected usage rate
- a customer-obligated term regulates then the usage rate (#1)
- provider-obligated terms specify availability and response time (#2, #3)
- provider-obligated action increases the customer-specific storage capacity whenever actual utilization goes beyond 90% (#4)

Examples (in pseudo concrete syntax)

```
<identifiableElement xsi:type="service:Service">
...
 <contextVariables>

  <!-- variable for relative uptime, default: 90% -->
  <variableDeclaration xsi:id="varCustUptime">
   <name>
    <value> customRelativeUptime </value>
    <type> name </type>
   </name>
   <defaultValue> 90 </defaultValue>
   <typeReference>
    <classificationSystemID> http://www.internet-of-services.com/serviceTypes </classificationSystemID>
    <classID> percent </classID>
    <unitSymbol> % </unitSymbol>
    <descriptions>
     <description>
      <value> percentage </value>
      <type> freetextShort </type>
      <language> en </language>
     </description>
    </descriptions>
   </typeReference>
  </variableDeclaration>

  <!-- variable for usage rate (throughput), default: 100 requests per second -->
  <variableDeclaration xsi:id="varUsageRate">
   <name>
    <value> usageRate </value>
    <type> name </type>
   </name>
   <defaultValue> 100 </defaultValue>
   <typeReference>
    <classificationSystemID> http://www.internet-of-services.com/serviceTypes </classificationSystemID>
    <classID> requests_per_second </classID>
    <unitSymbol> #/s </unitSymbol>

    <descriptions>
     <description>
      <value> number of requests per second </value>
      <type> freetextShort </type>
      <language> en </language>
     </description>
    </descriptions>
   </typeReference>
```

```
  </variableDeclaration>
</variables>

<serviceLevelProfiles>
 <serviceLevelProfile>

  <serviceLevels>

   <!-- service level #1 -->
   <serviceLevel xsi:type="servicelevel:GuaranteedState">
    <!-- customer is obligated -->
    <obligatedParty> cust987 </obligatedParty>
    <descriptions>
     <description>
      <value> The maximum usage rate is below the agreed value. </value>
      <type> freetextShort </type>
      <language> en </language>
     </description>
    </descriptions>

    <stateSpecification>
     <!-- measured throughput is less than or equal to what is set as "usage rate" in the service context -->
     <value> metric["m_throughput"] <= variable["v_usageRate"] </value>
     <languageID> urn:example:expression_language </languageID>
     <attributes>
      <!-- reference to the variable -->
      <attribute xsi:id="v_usageRate" xsi:type="servicelevel:VariableReference">
       <reference> varUsageRate </reference>
      </attribute>
      <!-- metric for measuring throughput  -->
      <attribute xsi:id="m_throughput" xsi:type="slbaseext:TimePerformanceMetric">
       <relatesTo> intf3245 </relatesTo>
       <timePerformanceType> Throughput </timePerformanceType>
       <typeReference>
        <classificationSystemID>
          http://www.internet-of-services.com/serviceTypes
       </classificationSystemID>
        <classID> requests_per_second </classID>
        <unitSymbol> #/s </unitSymbol>
       </typeReference>
       <assessment>
        <value> service requests measured as number of Web service calls </value>
        <type> freetextLong </type>
        <language> en </language>
       </assessment>
      </attribute>
     </attributes>
    </stateSpecification>
   </serviceLevel>

   <!-- service level #2 -->
   <serviceLevel xsi:type="servicelevel:GuaranteedState">
    <!-- provider is obligated -->
    <obligatedParty> prov321 </obligatedParty>

    <descriptions>
     <description>
      <value>
```

```
        The relative uptime of the service is at least what has been specified by the customer.
       </value>
      <type> freetextShort </type>
      <language> en </language>
     </description>
    </descriptions>

    <stateSpecification>
     <!-- measured relative uptime is greater than or equal to
     what is set in the service context (cust. input)-->
     <value> metric[“m_relUptime”] >= variable[“v_custUptime”] </value>
     <languageID> urn:example:expression_language </languageID>

     <attributes>
      <!-- reference to the variable -->
      <attribute xsi:id=”v_custUptime” xsi:type=”servicelevel:VariableReference”>
       <reference> varCustUptime </reference>
      </attribute>
      <!-- metric for measuring relative uptime in the last hour -->
      <attribute xsi:id=”m_relUptime” xsi:type=”slbaseext:TimePerformanceMetric”>
       <timePerformanceType> UptimeRelative </timePerformanceType>
       <typeReference>
        <classificationSystemID>
          http://www.internet-of-services.com/serviceTypes
        </classificationSystemID>
        <classID> percent </classID>
        <unitSymbol> % </unitSymbol>
        <descriptions>
         <description>
          <value> percentage </value>
          <type> freetextShort </type>
          <language> en </language>
         </description>
        </descriptions>
       </typeReference>
       <assessment>
        <value> relative uptime measured as the percentage of absolute uptime in the last hour </value>
        <type> freetextLong </type>
        <language> en </language>
       </assessment>
      </attribute>
     </attributes>
    </stateSpecification>

   </serviceLevel>

   <!-- service level #3 -->
   <serviceLevel xsi:type=”servicelevel:GuaranteedState”>
    <!-- provider is obligated -->
    <obligatedParty> prov321 </obligatedParty>

    <descriptions>
     <description>
      <value> The response time of each service invocation is below 2 seconds. </value>
      <type> freetextShort </type>
      <language> en </language>
     </description>
    </descriptions>
```

```
 <stateSpecification>

  <!-- measured end-to-end service time is less than 2000 milliseconds -->
  <value> metric[“m_e2eTime”] < constant[“c7”] </value>
  <languageID> urn:example:expression_language </languageID>

  <attributes>
   <!—end-to-end service time threshold (modeled as constant) -->
   <attribute xsi:id=”c7” xsi:type=”servicelevel:Constant”>
    <value> 2 </value>
    <typeReference>
     <classificationSystemID>
      http://www.internet-of-services.com/serviceTypes
     </classificationSystemID>
     <classID> duration_second </classID>
     <unitSymbol> s </unitSymbol>
     <descriptions>
      <description>
       <value> duration in seconds </value>
       <type> freetextShort </type>
       <language> en </language>
      </description>
     </descriptions>
    </typeReference>
   </attribute>
   <!-- metric for measuring end-to-end service time-->
   <attribute xsi:id=”m_e2eTime” xsi:type=”slbaseext:TimePerformanceMetric”>
    <timePerformanceType> EndToEndTime </timePerformanceType>
    <typeReference>
     <classificationSystemID>
       http://www.internet-of-services.com/serviceTypes
     </classificationSystemID>
     <classID> duration_millisecond </classID>
     <unitSymbol> ms </unitSymbol>
     <descriptions>
      <description>
       <value> duration in milliseconds </value>
       <type> freetextShort </type>
       <language> en </language>
      </description>
     </descriptions>
    </typeReference>
    <assessment>
     <value> end-2-end service time measured by provider for each request </value>
     <type> freetextLong </type>
     <language> en </language>
    </assessment>
   </attribute>
  </attributes>

 <stateSpecification>
</serviceLevel>

<!-- service level #4 -->
<serviceLevel xsi:type=”servicelevel:GuaranteedAction”>
 <!-- provider is obligated -->
 <obligatedParty> prov321 </obligatedParty>
```

```
        <descriptions>
          <description>
            <value> Whenever storage utilization of service instance exceeds 90 %, storage is scaled up. </value>
            <type> freetextShort </type>
            <language> en </language>
          </description>
        </descriptions>

        <preconditionSpecification>
          <!-- measured storage utilization is greater than 90 % -->
          <value> metric[“m_storageUtil”] > constant[“c8”] </value>
          <languageID> urn:example:expression_language </languageID>

          <attributes>
            <!-- customer-specific storage utilization threshold (modeled as constant) -->
            <attribute xsi:id=”c8” xsi:type=”servicelevel:Constant”>
              <value> 90 </value>
              <typeReference> <!-- same type description as specified in metric below --> </typeReference>
            </attribute>
            <!-- metric for measuring customer-specific storage utilization -->
            <attribute xsi:id=”m_storageUtil” xsi:type=”servicelevel:Metric”>
              <typeReference>
                <classificationSystemID>
                    http://www.internet-of-services.com/serviceTypes
                </classificationSystemID>
                <classID> percent </classID>
                <unitSymbol> % </unitSymbol>
              </typeReference>
              <assessment>
                <value>
                  ratio between storage space used by the service for a particular customer and space allocated
                  to that customer, measured every hour
                </value>
                <type> freetextLong </type>
                <language> en </language>
              </assessment>
            </attribute>

          </attributes>
        <preconditionSpecification>

        <actionSpecification>
          <value> <!-- acquire and allocate an additional 25% of initially allocated storage space --> </value>
          <languageID> urn:example:action_language </languageID>
        </actionSpecification>

      </serviceLevel>
    </serviceLevels>
  </serviceLevelProfile>
 </serviceLevelProfiles>
...
</identifiableElement>
```