Project Acronym:	**OPTIMIS**
Project Title:	**Optimized Infrastructure Services**
Project Number:	**257115**
Instrument:	**Integrated Project**
Thematic Priority:	**ICT-2009.1.2 – Internet of Services, Software and Virtualisation**

ID2.2.2 Service Manifest Specification

Activity 2:

WP 2.2: *Service Manifest*

Due Date:		
Submission Date:		
Start Date of Project:		01/06/2010
Duration of Project:		
Organisation Responsible for the Deliverable:		SCAI
Version:		0.1
Status		Draft
Author(s):	Angela Rumpl Hassan Rasheed Oliver Waeldrich	SCAI
Reviewer(s)		

Table of Contents

1 Introduction

There are three steps in the OPTIMIS service lifecycle: construction of the service, deployment of the service, and operation of the service. Services are developed, orchestrated, and configured by SPs for deployment on IPs. The SP writes specification and configuration of the service manifest describing the functional and non-functional parameters of the service. Information relevant to the service manifest includes: VM images, thresholds for TREC factors the SP requests, location and cost constraints, capacity and elasticity requirements, KPIs to monitor, etc.

The OPTIMIS service manifest basically describes the requirements of the service provider for an infrastructure service provisioning process. The Service Manifest is therefore an abstract definition of the infrastructure services as expected by the service provider. It is possible to specify multiple infrastructure services in one manifest file. All aspects of these infrastructure services must be described in detail in the manifest. Figure 1 service manifest structure shows a high level overview of the service manifest structure.

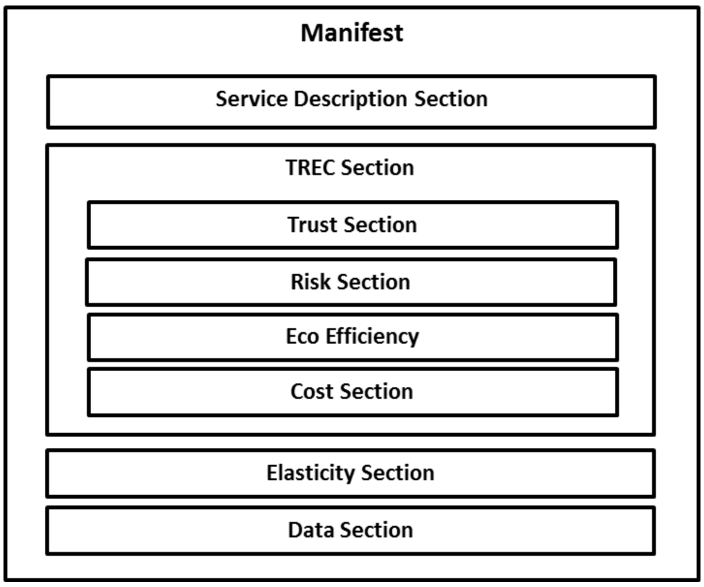

Figure 1 service manifest structure

The remainder of the document is structured as follows; first we give a high level overview of the manifest component model and we describe how references to service components are realized in the service manifest. Then we give an overview how the manifest can be extended with additional information in a particular domain, i.e. the service provider domain or the infrastructure provider domain. Next we provide a detailed description of the data types used in the service manifest in detail.

2 The Component Model

Applications that are deployed in the cloud often consist of more than one component; a web-application for example may require a web server, an application server and a database server to run. In order to perform efficiently, each application component may have different requirements in terms of numbers of allocated instances, CPU speed, memory, etc. Therefore, the OPTIMIS service provider must be able to describe multiple components of a service that it wishes to run along with the constraints for each component in a single service manifests.

The service manifest supports this requirement by a simple component model. Basically, each service may consist of one or more components. A service is described in the service description section of the service manifest. The components of the service are specified in this section as *Service Component* elements. Each service component must be identified by a component id which must be unique within a single manifest instance.

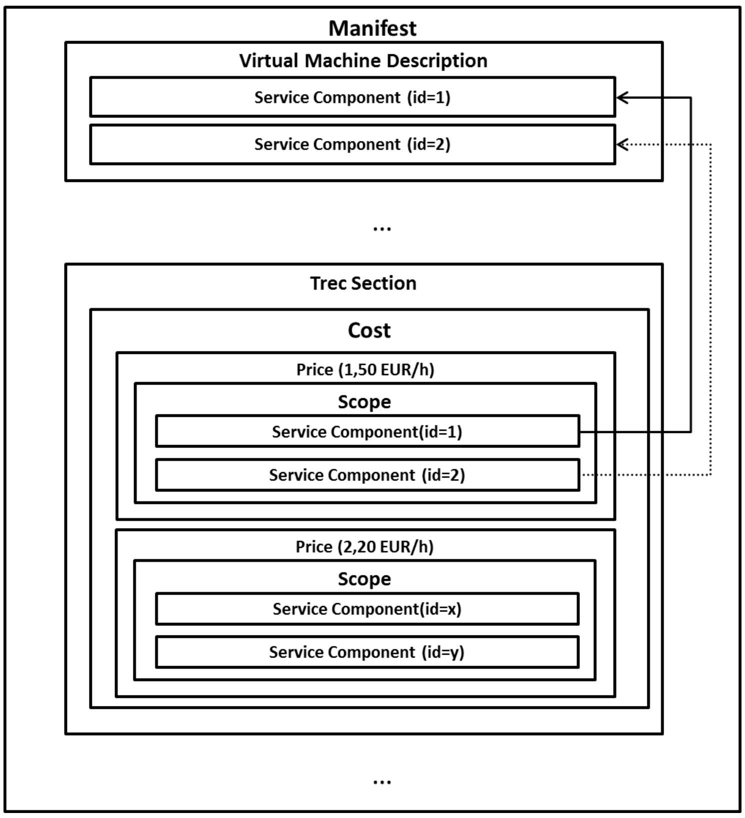

Figure 2: example of a service component reference

Service components must be referable by other sections in the service manifest. It must for example be possible to specify a different price for the different components of a service. Therefore the different sections in the manifest may specify a component scope. In general we distinguish between two types of sections in the manifest: *global sections* and *scoped sections*. Global sections apply to the general service provisioning process, in other words these sections apply to the aggregated service. Scoped sections apply to at least one service component, but they may refer to multiple service components. If a scoped section refers to a particular service component, it must refer to a valid service component id. Entities that process the service manifest must take these references into account. Figure 2 illustrates this behavior on the example of the cost section.

3 Annotating the Service Manifest

By default the service manifest only contains the information that is exchanged between service provider and infrastructure provider, i.e. a description of the different types of virtual machines that should be provided by the infrastructure provider, the number of instances that can be allocated for each VM type, the trust, risk, eco efficiency and cost parameters, etc. This information is usually insufficient at multiple stages of the service provisioning process. The service provider for example wants to specify software dependencies for a particular service component in the manifest. A set of contextualization tools allow the service provider to install these software dependencies into a bare VM image offered by the infrastructure provider. The resulting customized VM image is then uploaded to the infrastructure provider and used for the service provisioning process. Another example is the provisioning of the requested VMs in the infrastructure provider domain. After the service manifest was received by the infrastructure provider the IP must generate a document which describes all VM instances for all components defined in the service manifest service description section. This document must be passed along with the service manifest. Multiple components need to process the VM instance description document for different purposes, i.e. the Cloud Optimizer needs to specify which VM instance is deployed on which host, the IP contextualization tool adopt for example the network configuration of the virtual machine to sensible values with respect of the allocated infrastructure, data management might change the references to the VM images according to the physical file names of the images which were transferred to the execution system. These examples already show that there is a need to specify a wide set of information in the service manifest. In general we can distinguish three categories of information:

1. Information only used in the service provider domain
2. Information only used in the infrastructure provider domain
3. Shared information

The shared information is basically what is defined in the core service manifest. This information is passed from the SP to the IP when the service is contracted, i.e. when the SLA for a service provisioning process is created. Information that is only used in one of these domains, i.e. for preparing a VM image of for deploying and configuring a service, can be included in the service manifest as an extension document. This allows passing this type of information between the different services within one domain.

Note: Extension documents are not passed between domains. They are not part of the contract between service provider and infrastructure provider. Components from other domains than the one for which the extension document was defined are completely unaware of the existence of this information. This is also true for different infrastructure providers, for example in a cloud bursting scenario.

This specification defines two types of extension documents, one for infrastructure providers and one for service providers. Each extension document is defined in a separate namespace. The definition of the namespaces is provided below.

Namespaces

Service Provider Namespace	*http://schemas.optimis.eu/optimis/sp-extensions*
Infrastructure Provider Namespace	*http://schemas.optimis.eu/optimis/ip-extensions*

As mentioned before, the domain-specific extension documents are included in the service manifest and can therefore be passed along with the manifest between the services of one domain. For that purpose the service manifest defines the appropriate extension points (xs:any elements, see XML Schema specification). Figure 3 illustrates the extension mechanism on the example of an infrastructure provider extension.

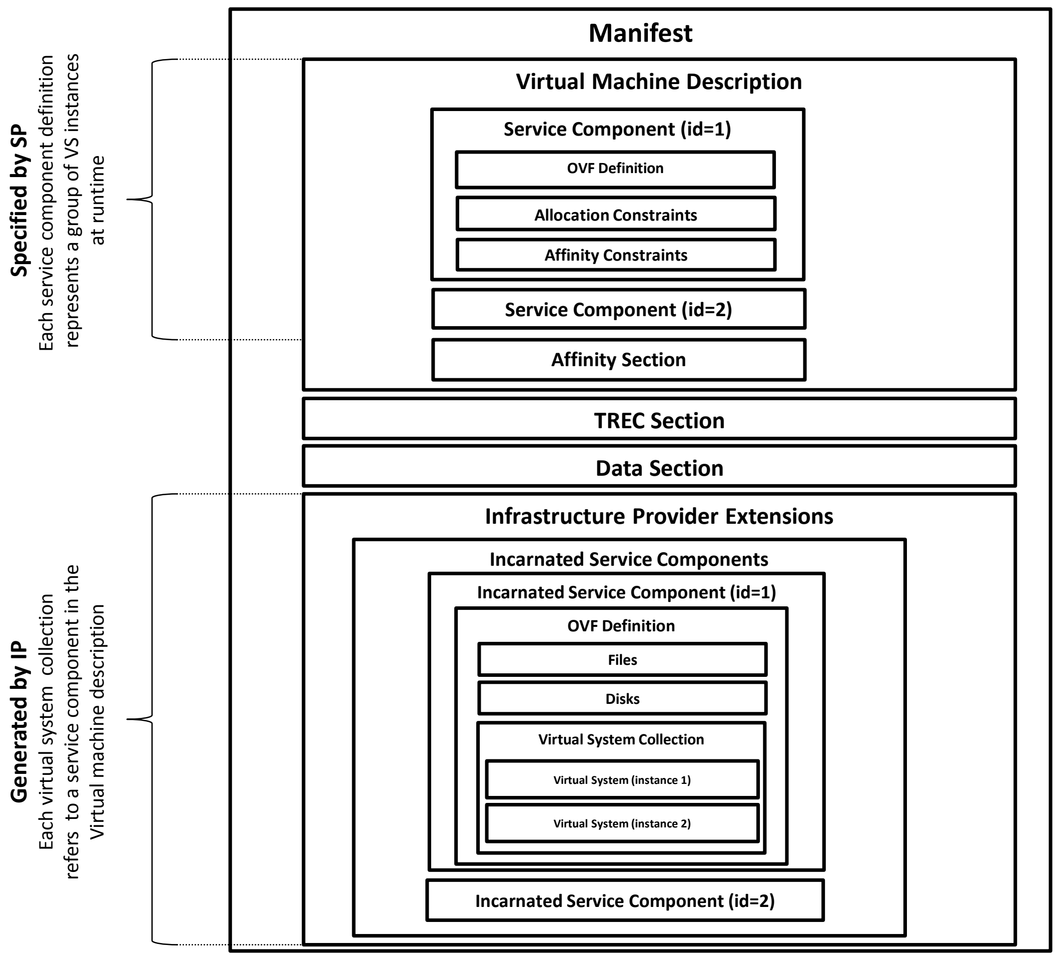
Specified by SP
Each service component definition represents a group of VS instances at runtime
Generated by IP
Each virtual system collection refers to a service component in the Virtual machine description
Manifest
Virtual Machine Description
Service Component (id=1)
OVF Definition
Allocation Constraints
Affinity Constraints
Service Component (id=2)
Affinity Section
TREC Section
Data Section
Infrastructure Provider Extensions
Incarnated Service Components
Incarnated Service Component (id=1)
OVF Definition
Files
Disks
Virtual System Collection
Virtual System (instance 1)
Virtual System (instance 2)
Incarnated Service Component (id=2)

Figure 3: example of an infrastructure provider extension in the manifest

4 Manifest Splitting

For several scenarios it is required to extract a component from a manifest and have two individual manifests to be able to split the deployment of virtual machines across several data centers.

The splitting process follows a simple algorithm:

- Provide the componentId and the maximum number of instances to be extracted.
- Check that federation is allowed for this manifest
- Check that the affinity of this component is Low which means that instances of this component are allowed to be transferred to other data centers.
- Check that the provided maximum number of instances is smaller or equal to the instances in the new manifest.
    - Equal: Cut the whole VirtualMachineComponent from the old manifest and paste it to the new manifest
    - Smaller: Copy and paste the VirtualMachineComponent element to the new manifest; remove the provided number of instances from the allocation constraints in the original and set the number of instances in the new manifest to the provided number of instances.
-

5 Service Manifest Schema

The following section gives a detailed definition of the types defined for the service manifest schema. It defines the main purpose of the different sections of the manifest and describes the use of the elements within these sections.

5.1 Service Description Section

This section provides a definition of an OPTIMIS Service. A service MAY consist of multiple components. At least one service component MUST be defined in the service description section. In principle different types of service components can be defined for a manifest. By default this specification defines a virtual machine service component that can be used to deploy one particular virtual system in an OPTIMIS IP infrastructure (see Virtual Machine Description).

```
<opt:ServiceDescriptionSection>
    <opt:ServiceComponent componentId=”xs:string”> +
</opt:ServiceDescriptionSectionType>
```

/opt:ServiceDescriptionSection/opt:ServiceComponent

This element describes one component of a service that is deployed in the OPTIMIS infrastructure; i.e. one type of VMs with a given number of instances. At least one service component element MUST be specified in a service description section.

/opt:ServiceDescriptionSection/opt:ServiceComponent/@opt:componentId

The REQUIRED attribute “*componentId*” represents the ID of the service component. The id MUST be unique in the service manifest. It is used to link other sections in the service manifest with one particular service component, i.e. in order to associate specific costs with that component.

5.2 Virtual Machine Description

A virtual machine description is a specific representation of an OPTIMIS service description. It is used to describe a set of virtual machine components that are deployed in an OPTIMIS IP infrastructure. It also describes the affinity of the components.

```
<opt:VirtualMachineDescription>
   <opt:VirtualMachineComponent>
        opt:VirtualMachineComponent
   </opt:VirtualMachineComponent>
   <opt:AffinitySection>
      opt:AffinitySection
   </opt:AffinitySection >
</opt:VirtualMachineDescription
```

5.2.1 Virtual Machine Component

A virtual machine component is a specific representation of an OPTIMIS service component. It is used to describe one particular class of virtual machines that are deployed in an OPTIMIS IP infrastructure. A class of virtual machines is defined as a set of virtual machine instances that are based on the same *virtual system definition*. A virtual system definition is provided as OVF description. This description MUST define exactly one virtual system. Virtual system collection is NOT supported.

The virtual machine description section (Figure 4) consists of a number of service components that together form a service described in the manifest.

Each service component comprises an OVF description section, an allocation constraints section and an affinity section.

- The *OVFDefinition* element (see 5.2.1.1) is a template for creating instances of this component at the IP site. It provides information on location, format, network connection and virtual system description to be used for creating component instances.
- The *AllocationConstraints* element (see 5.2.1.2) is used to define the maximum and minimum number of component instances.
- *AffinityConstraints* (see 5.2.1.3) describe the level of affinity the incarnated instances must have.

For further information on how the component instances are created see section 6 on IP Extensions.

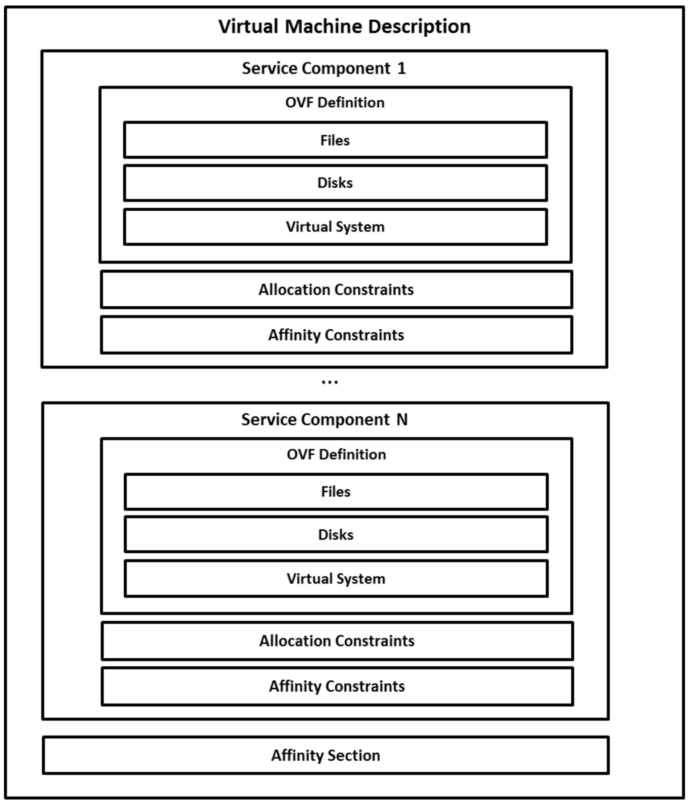

Figure 4: virtual machine description

5.2.1.1 OVF Definition

The OPTIMIS virtual machine service component is described in the Open Virtualization Format (OVF). The OVF specification is a standard being developed within the Distributed Management Task Force (DMTF) association to promote an open, secure, portable, efficient, and extensible format for the packaging and distribution of software to be run in virtual machines.

The service component description MUST provide only the following elements:

- A list of file references to all external files that are part of the OVF package, defined by the *References* element and its *File* child elements. These are typically virtual disk files, ISO images, and internationalization resources. The specified files refer to the appropriate files in the OPTIMIS VM repository. The IP uses this information at runtime to stage in the referenced files to the allocated execution systems.
- A *DiskSection* that describes meta-information about virtual disks in the OVF package. Virtual disks and their metadata are described outside the virtual hardware to facilitate sharing between virtual machines within an OVF package. A virtual disk MUST link to a file definition in the references section.
- A description of the virtual machine, more specifically a *VirtualSystem* element. The *VirtualSystem* definition MUST be contained in a valid OPTIMIS service component description. Only one virtual system is allowed. Virtual System Collections are NOT supported in the service component definition.

5.2.1.2 Allocation Constraints

In this section we define how many instances of a component can be created. An allocation constraint is directly associated with a particular service component. It defines the boundaries of the virtual system instances that can be started for a particular service component.

/opt:AllocationConstraintType/opt:LowerBound

The lower bound defines the minimum number of VM instances that must be provided. The infrastructure provider should not allocate less VM instances as defined by the lower bound.

/opt:AllocationConstraintType/opt:UpperBound

The upper bound defines the maximum number of VM instances that can be provided. The infrastructure provider must not allocate more VM instances as defined by this bound.

/opt:AllocationConstraintType/opt:Initial

The initial number of instances that are allocated

5.2.1.3 Affinity Constraints

This describes the level of affinity between incarnated instances of a service component. A service provider may specify one of the following values: *Low*, *Medium* or *High*. The values have the following definition:

Low: The service component instances don't impose special deployment or communication restrictions. Therefore the component instances can be distributed across different data centers.

Medium: The service component instances impose a medium coupling with respect to deployment or communication. Therefore the component instances must be deployed within one data center.

High: The service component instances impose high restrictions with respect to deployment or communication. Therefore the component instances must be deployed within one cluster.

5.2.2 Affinity Section

```
<opt:AffinitySectionType>
   <opt:Rule>
       <opt:Scope>
            opt:ScopeArray
       </opt:Scope>
       <opt:AffinityLevel>
               {  HIGH | MEDIUM | LOW  }
        </opt:AffinityLevel>
    </opt:Rule> +
</opt:AffinitySectionType
```

In the OPTIMIS service manifest there can be two levels of affinity. First the affinity between a service component's instances, as described in section 5.2.1.3, and second the affinity between service components. This section describes the level of affinity between service components. It therefore uses the scope array to make a reference to the service components the affinity level applies to.

opt:Rule/opt:Scope/opt:ScopeArray

Specifies a set of components on which a specific affinity level is applied. The components are identified by their component ids.

opt:Rule/opt:AffinityLevel

A service provider may specify one of the following values: *Low*, *Medium* or *High*. The values have the following definition:

Low: The service components don't impose special deployment or communication restrictions. Therefore the components can be distributed across different data centers.

Medium: The service components impose a medium coupling with respect to deployment or communication. Therefore the components must be deployed within one data center.

High: The service components impose high restrictions with respect to deployment or communication. Therefore the components must be deployed within one cluster.

5.3 TREC Section

```
<opt:TRECSectionType>
    <opt:TrustSection>
        opt:TrustSection
    </opt:TrustSection>  ?
    <opt:RiskSection>
        opt:RiskSection
    </opt:RiskSection>  ?
    <opt:EcoEfficiencySection>
        opt:EcoEfficiencySection
    </opt:EcoEfficiencySection>  ?
    <opt:CostSection>
        opt:CostSection
    </opt:CostSection>  ?
</opt:TRECSectionType>
```

5.3.1 Trust

```
<opt:TrustSectionType>
    <opt:TrustLevel>
        opt:TrustLevel
    </opt:TrustLevel>
    <xsd:any namespace="##other">  *
</opt:TrustSectionType>
```

Specifies the OPTIMIS trust parameters in a TREC section.

/opt:TrustLevelType/xsd:int

Specifies the OPTIMIS trust level that is used for delegation in a federated cloud scenario

5.3.2 Risk

```
<opt:RiskSectionType>
    <opt:RiskLevel>opt:RiskLevel</opt:RiskLevel>  ?
    <opt:AvailabilityArray>
        <opt:Availability>
            <opt:Availability AssessmentInterval="xsd:duration">
                 xs:int
            </opt:Availability>
        </opt:Availability>  *
    </opt:AvailabilityArray>  ?
    <xsd:any namespace="##other">  *
</opt:RiskSectionType>
```

/opt:RiskLevelType/xsd:int

Specifies the OPTIMIS risk level that is used for delegation in a federated cloud scenario

/opt:RiskSectionType/opt:AvailabilityArray

Contains one or more Availability elements, each availability element specifies the availability of a virtual machine in a given assessment interval.

/opt:AvailabilityArrayType/opt:Availability

Specifies the guaranteed availability of a virtual machine in a given assessment interval

/opt:AvailabilityType/@opt:assessmentInterval

Defines the duration of the assessment interval in seconds

5.3.3 EcoEfficiency

```
<opt:EcoEfficiencySectionType>
   <opt:LEEDCertification>
       opt:LEEDCertificationConstraint
   </opt:LEEDCertification> ?
   <opt:BREEAMCertification>
       opt:BREEAMCertificationConstraint
   </opt:BREEAMCertification> ?
   <opt:EuCoCCompliant>xsd:boolean</opt:EuCoCCompliant> ?
   <opt:EnergyStarRating>opt:EnergyStarRating</opt:EnergyStarRating> ?
   <xsd:any namespace="##other">  *
</opt:EcoEfficiencySectionType>
```

The following describes the attributes and elements listed in the schema above:

/opt:EcoEfficiencySectionType/opt:LEEDCertificationConstraintType

Provides enumerated values as follows:

`NotRequired, Certified, Silver, Gold, Platinum`

/opt:EcoEfficiencySectionType/opt:BREEAMCertification

Provides enumerated values as follows:
NotRequired, Pass, Good, VeryGood, Excellent, Outstanding

/opt:EcoEfficiencySectionType/opt:EuCoCCompliant

Specifies Boolean value

/opt:EcoEfficiencySectionType/opt:EnergyStarRating

Specifies range value between 1 and 100

/opt:EcoEfficiencySectionType/{any##other}

5.3.4 Cost

```
<opt:CostSection>
      <opt:PricePlan opt:currency="xs:string" opt:planCap="0.0" opt:planFloor="0.0">
        <opt:Scope>
          <opt:componentId>opt:componentId</opt:componentId>
        </opt:Scope>
        <opt:PlanComponents>
          <opt:PriceComponent opt:componentCap="0.0" opt:componentFloor="0.0">
```

```
            <opt:ComponentLevels>
              <opt:PriceLevel>
                <opt:AbsoluteAmount>0.0</opt:AbsoluteAmount>
                <opt:PriceMetrics>
                  <opt:PriceMetric opt:factor="0.0">opt:PriceMetric</opt:PriceMetric>
                </opt:PriceMetrics>
                <opt:LevelFences>
                  <opt:PriceFence>
                    <opt:BusinessTerm>opt:BusinessTerm</opt:BusinessTerm>
                       <opt:BusinessTermExpression>
                          opt:BusinessTermExpression
                       </opt:BusinessTermExpression>
                       <opt:QuantityLiterals>
                          <opt:Quantity opt:id="xs:string">
                            <opt:Amount>0.0</opt:Amount>
                            <opt:TypeReference>opt:TypeReference</opt:TypeReference>
                          </opt:Quantity>
                       </opt:QuantityLiterals>
                  </opt:PriceFence>
                </opt:LevelFences>
              </opt:PriceLevel>
            </opt:ComponentLevels>
            <opt:Multiplier>opt:Multiplier</opt:Multiplier>
          </opt:PriceComponent>
        </opt:PlanComponents>
      </opt:PricePlan>
    </opt:CostSection>
```

In OPTIMIS, cost is an explicit parameter throughout the full service lifecycle. The OPTIMIS tools will incorporate economics-related features and thus will e.g., facilitate comparisons of alternative configurations for a service, giving rise to cost efficient services. These terms and examples are adopted from the Unified Service Description Language (USDL) Pricing Module.

/opt:CostSection/opt:PricePlan

A PricePlan is a set of charges associated with a network-provisioned entity. Alternative sets of fees (i.e. alternative PricePlans) of the same service provision may be made available for the consumer to choose from, for example to offer the consumer the choice between a flat price scheme and a usage-based scheme (a common practice in the telecommunication industry).
Several PricePlans may exist for the same service in order to suit different user profiles and charge them appropriately (e.g. heavy- and light-usage users), or as a key price customization instrument to individually match diverse service valuations. There are three attributes associated with the PricePlan term.

/opt:CostSection/opt:PricePlan/@opt:currency

as a name string, EString: the currency for all price amounts within this PricePlan, e.g. , EUO.

/opt:CostSection/opt:PricePlan/@planCap

as a float num., EFloat: providing this maximum PricePlan value prevents from charging the user a higher total price, regardless of the cumulative total price the components and adjustments within this PricePlan may eventually amount to. Example: A cap may be used to set an upper limit in a strictly usage-based plan.

/opt:CostSection/opt:PricePlan/@planFloor

as a float num., EFloat: providing this minimum PricePlan value prevents from charging the user a lower total price, regardless of the cumulative total price the components and adjustments within this PricePlan may eventually amount to. Example: A floor may be used to set a lower limit to discounts that may result in an excessively low price.

/opt:CostSection/opt:PricePlan/opt:PriceComponent

PriceComponents are fees included in a PricePlan, which subject to conditions (expressed as PriceFences) may contribute to the total amount charged. Components within the same plan are summed together in order to get the total amount (price of the service). Common examples of PriceComponents that may coexist in the same PricePlan are: startup or membership charges (to access the service), periodic subscription fees (with a certain recurrence - e.g. monthly - as long as committed to by the contract), pay-per-unit charges (whose total will be proportional to the metered usage), options or feature dependent charges. The final value of the component will depend on the active PriceLevel (determined by the evaluation of the relative PriceFences) and the PriceAdjustments that may apply (e.g. discounts). There are two attributes associated with the PriceComponent term:

/opt:CostSection/opt:PricePlan/opt:PriceComponent/@opt:componentCap

as a float num., EFloat: providing this maximum PriceComponent value prevents the component final price from exceeding a certain amount, regardless of its levels and the parameters they are indexed to. Example: A cap may be used to set an upper limit for a component whose levels vary with usage.

/opt:CostSection/opt:PricePlan/opt:PriceComponent/@opt:componentFloor

componentFloor, as a float num., EFloat: providing this minimum PriceComponent value prevents the component final price from falling below a certain amount, regardless of its levels and the parameters they are indexed to. Example: A floor may be used to set a lower limit for a component whose levels vary with usage.

/opt:CostSection/opt:PricePlan/opt:PriceComponent/opt:PriceLevel

PriceLevel captures amounts charged by a PriceComponent. Since each PriceComponent may assume several values depending on the provider's price segmentation strategies, it is allowed to contain multiple PriceLevels. This allows shaping charged amounts according to customers' behavior and aligning usage with capacity or incurred costs (just like utilities do by offering different electricity rates for different times of day).

/opt:CostSection/opt:PricePlan/opt:PriceComponent/opt:PriceLevel/opt:PriceMetric

PriceMetric represents the unit of measurement by which the customer is charged for the consumption of the service or bundle. Metrics can be abstract/un-typed (e.g. per invocation) or typed (e.g. per MByte). The latter are covered by the sub-class TypedPriceMetric. The attributes that defines the PriceMetric is the factor, as a float num., EFloat: the minimum block of units that is priced, i.e. the step increase the price metric may take. It may also be a fraction. Examples: - A Gigabyte metric could be expressed equivalently as a Megabyte metric with a factor 1024 - A professional service priced with hourly rates but charged in 15 minutes increments (factor would be 0.25).

/opt:CostSection/opt:PricePlan/opt:PriceComponent/opt:PriceLevel/opt:TypedPriceMetric

TypedPriceMetric represents a concretely typed price metric, i.e. a metric associated with a defined unit of measurement. It is defined by one attribute:

/opt:CostSection/opt:PricePlan/opt:PriceComponent/opt:ComponentLevels/opt:LevelFences/opt:PriceFence

PriceFence represents a conditional expression evaluated to determine if a price element (i.e. *PricePlan, PriceComponent* or *PriceLevel*) applies. Within a *PriceFence* a certain business entity (represented by the *businessTerm)* is compared to a certain value (or set of values - the literals available to account for the different dimensions of the service provision process).

5.4 Elasticity Section

```
<opt:ElasticitySection>
   <opt:VariableSet>opt:VariableSet</opt:VariableSet>
   <opt:ElasticityRules>opt:ElasticityRules</opt:ElasticityRules>
</opt:ElasticitySection>
```

5.4.1 Variable Set

```
<opt:VariableSet>
   <opt:Variable opt:name=”xs:string” opt:metric=”xs:string”
opt:type=”opt:ElasticityLocationTypeEnum”>
      <opt:Location>xs:string</opt:Location>
    </opt:Variable> +
</opt:VariableSet>
```

/opt:VariableSet

A variable set will contain the definition of variables used by elasticity rules to interpret the expression of a rule.

/opt:VariableSet/Variable

A variable contains all information necessary to locate its value.

/opt:VariableSet/Variable/@name

The name of the variable

/opt:VariableSet/ Variable/@metric

This specifies the metric of a variable, e.g. “int” if the variable represents an Integer value.

/opt:VariableSet/ Variable/@type

This element is used to identify the type of a variable. One of “internal” or “external” applies.

/opt:VariableSet/Variable/Location

The location must point to the variable’s value. This depends on the *type* attribute of the variable.

- **Internal:** the variable is defined inside the manifest document. The location of its value is therefore displayed as an xpath expression that points to a value inside the manifest.
  E.g. to locate the required availability value for an assessment interval of 1 day, we would use the following xpath:
  `”//opt:RiskSection/opt:AvailabilityArray/Availability[@opt:assessmentInterval=’P1D’]”`
- **External:** the variable’s value is the actual value provided by some monitoring interface. This location should be an URI which points to the monitored value.

5.4.2 Elasticity Rules

```
<opt:ElasticityRule>
   <opt:Scope>opt:ScopeArray</opt:Scope>
   <opt:Condition>
      <opt:Expression>xs:string</opt:Expression>
      <opt:AssessmentCriteria>
         <opt:Window>xs:duration</opt:Window>
         <opt:Frequency>xs:int</opt:Frequency>
      </opt:AssessmentCriteria>
   </opt:Condition>
   <opt:Effect>
      <opt:Importance>xs:int</opt:Importance>
      <opt:Action>xs:string</opt:Action>
   </opt:Effect>
</opt:ElasticityRule> +
```

The following describes the attributes and elements listed in the schema above:

/opt: ElasticityRule /@name

The name of the elasticity rule refers to a KPI and it must be unique.

/opt: ElasticityRule /opt:Scope

Specifies a set of VM instances on which a specific Elasticity Action is invoked. Specific elasticity actions are defined for a specific KPI. The elasticity actions are invoked for the VMs in the same sequence as specified in this list.

/opt:ElasticityRule/opt:Condition

The condition holds all necessary information to verify if a KPI is met and what effects should apply if the conditions are not fulfilled.

/opt:ElasticityRule/opt:Condition/opt:Expression

The expression will be interpreted by the elasticity component to verify if the condition is fulfilled or not. E.g.: “`REQ_AVAILABILITY le PROVIDED_AVAILABILITY`”

/opt:ElasticityRule/opt:Condition/opt:AssessmentCriteria

The assessment criteria specifies window and frequency of an assessment

/opt:ElasticityRule/opt:Condition/opt:AssessmentCriteria/opt:Window

Specifies the duration of the assessment

/opt:ElasticityRule/opt:Condition/opt:AssessmentCriteria/opt:Frequency

Specifies the frequency of an assessment

/opt:ElasticityRule/opt:Effect

The effect describes the consequences that will happen if an expression validates to false.

/opt:ElasticityRule/opt:Condition/opt:Effect/opt:Importance

The importance of the effect

/opt:ElasticityRule/opt:Condition/opt:Effect/opt:Action

The actual action, e.g. scale up/ scale down

5.5 Data Protection Section

```
<opt:DataProtectionSectionType>
    <opt:EligibleCountryList>
        <opt:Country>opt:ISO3166Alpha2</opt:Country> +
    </opt:EligibleCountryList>  ?
    <opt:NonEligibleCountryList>
        <opt:Country>opt:ISO3166Alpha2</opt:Country> +
    </opt:NonEligibleCountryList>  ?
    <opt:DataProtectionLevel>
        opt:DataProtectionLevel
    </opt:DataProtectionLevel>  ?
    <opt:DataEncryptionLevel>
        <opt:EncryptionAlgorithm>
        <opt:EncryptionKeySize>
        <opt:CustomEncryptionLevel>
    </opt:DataEncryptionLevel>  ?
    <xsd:any namespace="##other">  *
</opt:DataProtectionSectionType>
```

The following describes the attributes and elements listed in the schema above:

/opt:EncryptionLevelType/opt:EncryptionAlgorithmType

Enumeration values are: NotApplicable, AES, Twofish, AES-Twofish, AES-Twofish-Serpent, Serpent-AES, Serpent-Twofish-AES, Twofish-Serpent

/opt: EncryptionLevelType/opt:CustomEncryptionLevel

6 IP Extensions

The IP extension document contains additional information for deploying services in an OPTIMIS IP infrastructure based on the service specification provided by the service provider. The IP extension document is generated by the OPTIMIS IP before deploying the services requested by the SP. The IP extension document is dynamically included in the manifest and available for all services in the IP domain. It is passed along with the manifest and the different IP components can read the extension document and process its content during a service provisioning process. The IP extension document is only valid in the domain of the particular IP that generated the document; i.e. this information will not be passed to subcontractors such as other IPs in a cloud bursting scenario.

IP Extension Document Processing Model

The SP specifies the services it needs in order to run an application as service components in the manifest *Service Description Section*. Each service components acts as a prototype for the virtual system instances that are finally provided by the OPTIMIS IP; this means the OVF descriptions of the virtual systems that are finally deployed in the OPTIMIS IP infrastructure are derived from the abstract service component definitions. The number of virtual system instances that are provided per service component is defined in the allocation section of a service component.

The IP extensions document is an incarnation of the abstract services component definition specified in the manifest. It basically contains a set of *incarnated service definitions*. An incarnated service definition is an OVF document which describes all virtual systems that can be deployed for one service component. Each incarnated service refers to one service component definition in the manifest. An incarnated service is basically represented as an OVF document which contains a Virtual System Collection with the virtual systems that can be deployed for a service component with respect to the allocation constraints. Figure 5 shows the IP extension document which is the result of an incarnation process.

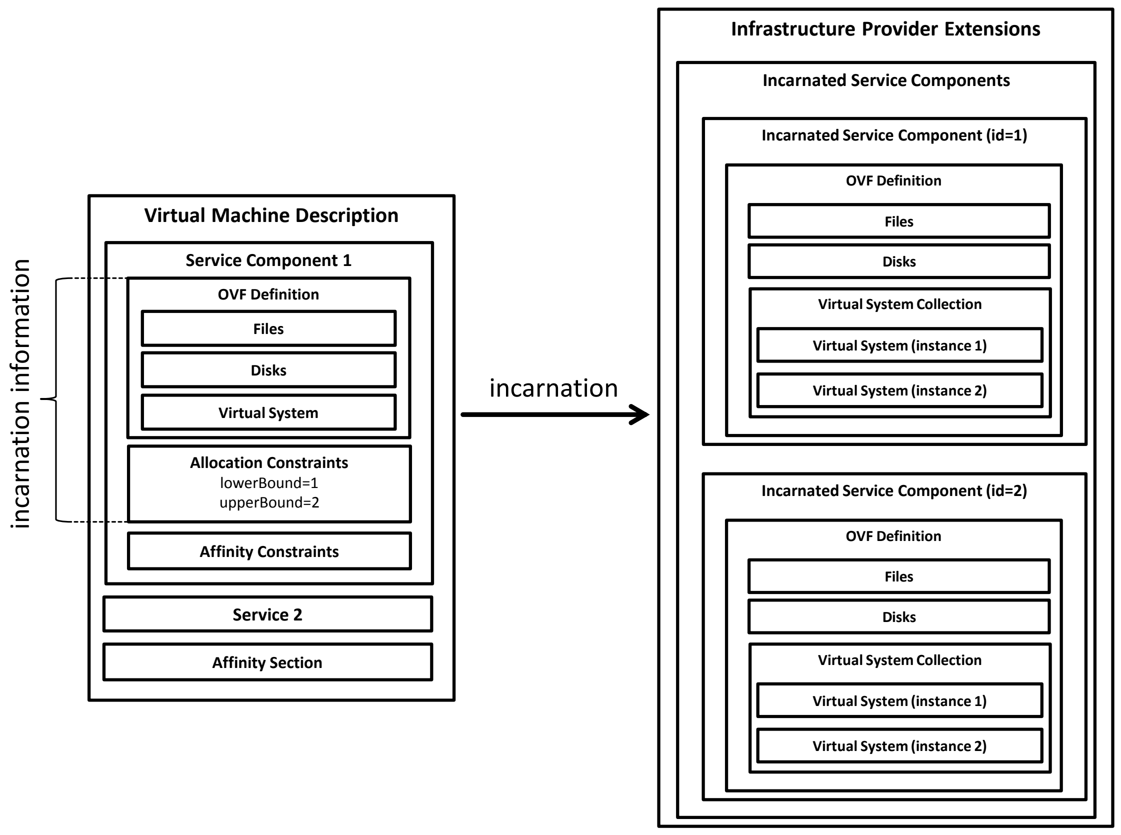
Virtual Machine Description
Service Component 1
OVF Definition
Files
Disks
Virtual System
Allocation Constraints
lowerBound=1
upperBound=2
Affinity Constraints
Service 2
Affinity Section
incarnation information
incarnation
Infrastructure Provider Extensions
Incarnated Service Components
Incarnated Service Component (id=1)
OVF Definition
Files
Disks
Virtual System Collection
Virtual System (instance 1)
Virtual System (instance 2)
Incarnated Service Component (id=2)
OVF Definition
Files
Disks
Virtual System Collection
Virtual System (instance 1)
Virtual System (instance 2)

Figure 5: generated extension document during the service incarnation process

Incarnation Model

The incarnation process is the process of generating a concrete OVF document from the abstract OVF template specified in the service component definition. The incarnation process uses the allocation constraints defined in the service component's allocation constraints section to determine the maximum number of instances of the service component. For each possible VM instance a new virtual system is generated in the resulting OVF file. The virtual systems are grouped in a Virtual System Group element. In order to successfully incarnate a service from the OVF template the template document must be valid, i.e. all reference in the template must be valid.

The incarnation process follows a simple algorithm illustrated below.

- Copy all references in the OVF template to the incarnated OVF
- Update the Ids of all copied references such as: newId = Id + "instance_" + i
- Copy all disks form the OVF template to the incarnated OVF
- Update the Ids of all copied disks such as: newId = Id + "instance_" + i
- Update the file references of all copied disks such as: newRef = Ref + "instance_" + i
- Copy the virtual system definition from the template to the incarnated virtual system group
- Update the Id of the copied virtual system such as: newId = Id + "instance_" + i
- Update the disk references of the copied virtual system such as: newRef = Ref + "instance_" + i

7 SP Extensions

```
<opt-sp:ServiceProviderExtensions>
  <opt-sp:VirtualMachineComponentConfiguration opt-sp:componentId="xs:string"> *
    <opt-sp:SoftwareDependencies>
      <opt-sp:Dependency> *
        <opt-sp:artifactId>xs:string</opt-sp:artifactId>
        <opt-sp:version>xs:string</opt-sp:version>
        <opt-sp:groupId>xs:string</opt-sp:groupId>
      </opt-sp:Dependency>
    </opt-sp:SoftwareDependencies>
    <opt-sp:SecurityKeyType>opt:SecurityKeyType</opt-sp:SecurityKeyType>
  </opt-sp:VirtualMachineComponentConfiguration>
</opt-sp:ServiceProviderExtensions>
```

The following describes the attributes and elements listed in the schema above:

/opt-sp:VirtualMachineComponentConfiguration

A "VirtualMachineComponentConfiguration" element is used to provide configuration settings for a specific virtual machine component. Exactly one configuration element per component can exist.

/opt-sp:VirtualMachineComponentConfiguration/@componentId

The componentId links the configuration with the respective VirtualMachineComponent in the VirtualMachineDescriptionSection.

/opt-sp:VirtualMachineComponentConfiguration/opt-sp:SoftwareDependencies

The software dependencies section holds a number of software dependencies required by this component at runtime.

/opt-sp:VirtualMachineComponentConfiguration/opt-sp:SecurityKeyType

The security key type specifies how the components vms can be accessed. Currently there are two options: SSH or VPN.

8 Appendix

8.1 Service Manifest

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:opt="http://schemas.optimis.eu/optimis/"
           xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
           targetNamespace="http://schemas.optimis.eu/optimis/"
           elementFormDefault="qualified" attributeFormDefault="qualified">
   <xs:import namespace="http://schemas.dmtf.org/ovf/envelope/1"

schemaLocation="http://schemas.dmtf.org/ovf/envelope/1/dsp8023_1.1.0.xsd"/>


   <xs:element name="ServiceManifest" type="opt:ManifestType">
      <!-- componentId is unique -->
      <xs:key name="vmComponentKey">
         <xs:selector xpath=".//opt:VirtualMachineComponent"></xs:selector>
         <xs:field xpath="@opt:componentId"></xs:field>
      </xs:key>
      <!-- elasticity rule names must be unique -->
      <xs:key name="elasticityRuleNameKey">
         <xs:selector xpath=".//opt:ElasticityRule"/>
         <xs:field xpath="@opt:name"/>
      </xs:key>
   </xs:element>
   <xs:element name="ElasticitySection" type="opt:ElasticitySectionType"/>
   <xs:element name="DataProtectionSection" type="opt:DataProtectionSectionType"/>
   <xs:element name="TRECSection" type="opt:TRECSectionType"/>

   <xs:element name="ServiceDescriptionSection"
               type="opt:ServiceDescriptionSectionType"/>
   <xs:element name="ServiceComponent" type="opt:ServiceComponentType"/>
   <xs:element name="VirtualMachineDescription"
               type="opt:VirtualMachineDescriptionType"
               substitutionGroup="opt:ServiceDescriptionSection"/>
   <xs:element name="VirtualMachineComponent" type="opt:VirtualMachineComponentType"
               substitutionGroup="opt:ServiceComponent"/>

   <xs:complexType name="ManifestType">
      <xs:annotation>
         <xs:documentation>
            Type definition of the OPTIMIS service manifest.
```

```
        </xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element ref="opt:ServiceDescriptionSection"/>
        <xs:element ref="opt:TRECSection" minOccurs="0"/>
        <xs:element ref="opt:ElasticitySection" minOccurs="0"/>
        <xs:element ref="opt:DataProtectionSection" minOccurs="0"/>
        <xs:any minOccurs="0" maxOccurs="unbounded" processContents="strict"
                namespace="##other"/>
    </xs:sequence>
    <xs:attribute name="manifestId" use="required">
        <xs:annotation>
            <xs:documentation>
                The manifest id is composed of the SLA name and SLA version. The
                values are separated by a colon.

                Remark: this pattern might change in future in case requirements
                change.
            </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:pattern value="\w[\w_\-]*\:\d+"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="serviceProviderId" type="xs:string"/>
</xs:complexType>

<xs:complexType name="ServiceDescriptionSectionType" abstract="true">
    <xs:annotation>
        <xs:documentation>
            Base type of an OPTIMIS Service Description. All service descriptions
            inherit from this type. Additional service descriptions MAY be defined
            for OPTIMIS and can be included into the service manifest as XSD
            substitution group.
        </xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element ref="opt:ServiceComponent" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="serviceId" type="xs:string" use="required"/>
    <xs:attribute name="isFederationAllowed" use="required" type="xs:boolean"/>
</xs:complexType>
```

```
<xs:complexType name="ServiceComponentType" abstract="true">
   <xs:annotation>
      <xs:documentation>
         Base type of an OPTIMIS Service Component. All service components
         inherit from this type. Additional service components MAY be defined for
         OPTIMIS and can be included into the service manifest as XSD
         substitution group.
      </xs:documentation>
   </xs:annotation>
   <xs:attribute name="componentId" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="AbstractVirtualMachineDescriptionType" abstract="true">
   <xs:annotation>
      <xs:documentation>
         Provisioning of plain virtual machines is the default OPTIMIS use case.
         The VirtualMachineServiceDescription specifies the VMs that are provided
         to a customer once an SLA is created.
      </xs:documentation>
   </xs:annotation>
   <xs:complexContent>
      <xs:extension base="opt:ServiceDescriptionSectionType">
         <xs:sequence>
            <!--<xs:element ref="opt:VirtualMachineComponent" minOccurs="1"-->
            <!--maxOccurs="unbounded"/>-->
            <xs:element name="AffinitySection" type="opt:AffinitySectionType"
                        maxOccurs="1"/>
            <xs:any namespace="##other" processContents="strict" minOccurs="0"
                    maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="VirtualMachineDescriptionType">
   <xs:annotation>
      <xs:documentation>
         Provisioning of plain virtual machines is the default OPTIMIS use case.
         The VirtualMachineServiceDescription specifies the VMs that are provided
         to a customer once an SLA is created.
      </xs:documentation>
   </xs:annotation>
   <xs:complexContent>
      <xs:restriction base="opt:AbstractVirtualMachineDescriptionType">
```

```
            <xs:sequence>
               <xs:element ref="opt:VirtualMachineComponent" minOccurs="1"
                           maxOccurs="unbounded"/>
               <xs:element name="AffinitySection" type="opt:AffinitySectionType"
                           maxOccurs="1"/>
               <xs:any namespace="##other" processContents="strict" minOccurs="0"
                       maxOccurs="unbounded"/>
            </xs:sequence>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>

   <xs:element name="OVFDefinition" type="ovf:EnvelopeType"/>
   <xs:element name="AllocationConstraints"
               type="opt:AllocationConstraintType"/>
   <xs:element name="AffinityConstraints" type="opt:AffinityConstraintType"/>
   <xs:complexType name="VirtualMachineComponentType">
      <xs:annotation>
         <xs:documentation>
            It is used to describe one particular class of virtual machines that are
            deployed in an OPTIMIS IP infrastructure.
         </xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="opt:ServiceComponentType">
            <xs:sequence>
               <xs:element ref="opt:OVFDefinition" maxOccurs="1"/>
               <xs:element ref="opt:AllocationConstraints" maxOccurs="1"/>
               <xs:element ref="opt:AffinityConstraints" maxOccurs="1"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>

   <xs:complexType name="AllocationConstraintType">
      <xs:annotation>
         <xs:documentation>
            Defines the scaling constraints for a specific component.
         </xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="LowerBound" type="xs:int"/>
         <xs:element name="UpperBound" type="xs:int"/>
         <xs:element name="Initial" type="xs:int"/>
```

```
    </xs:sequence>
  </xs:complexType>
  <!--
    Definition of OPTIMIS TREC parameters.
  -->
  <xs:complexType name="TRECSectionType">
    <xs:sequence>
      <xs:element name="TrustSection" type="opt:TrustSectionType" minOccurs="0"/>
      <xs:element name="RiskSection" type="opt:RiskSectionType" minOccurs="0"/>
      <xs:element name="EcoEfficiencySection" type="opt:EcoEfficiencySectionType"
                  minOccurs="0"/>
      <xs:element name="CostSection" type="opt:CostSectionType" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="TrustSectionType">
    <xs:annotation>
      <xs:documentation>
        Specifies the OPTIMIS trust parameters in a TREC section.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="MinimumTrustLevel" type="opt:TrustLevelType" minOccurs="0"/>
      <xs:element name="SocialNetworkingTrustLevel" type="opt:TrustLevelType"
minOccurs="0"/>
      <xs:element name="TrustLevel" type="opt:TrustLevelType"/>
      <xs:any namespace="##other" processContents="strict" minOccurs="0"
              maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="TrustLevelType">
    <xs:annotation>
      <xs:documentation>
        Specifies the OPTIMIS Trust Level that is used for delegation in a
        federated cloud scenario.

        TODO: is there a specification of the different Trust Levels in OPTIMIS?
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:int">
      <xs:minInclusive value="0"/>
    </xs:restriction>
  </xs:simpleType>
  <!--
    Definition of OPTIMIS Risk Constraints.
  -->
```

```
<xs:complexType name="RiskSectionType">
   <xs:sequence>
      <xs:element name="RiskLevel" type="opt:RiskLevelType" minOccurs="0"/>
      <xs:element name="AvailabilityArray" type="opt:AvailabilityArrayType"
                  minOccurs="0"/>
      <xs:any namespace="##other" processContents="strict" minOccurs="0"
              maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="AvailabilityArrayType">
   <xs:sequence>
      <xs:element name="Availability" type="opt:AvailabilityType" minOccurs="0"
                  maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="AvailabilityType">
   <xs:simpleContent>
      <xs:extension base="xs:double">
         <xs:attribute name="assessmentInterval" type="xs:duration"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
<xs:simpleType name="RiskLevelType">
   <xs:annotation>
      <xs:documentation>
         Specifies the OPTIMIS Risk Level that is used for delegation in a
         federated cloud scenario.

         TODO: see comment TrustLevelType.
      </xs:documentation>
   </xs:annotation>
   <xs:restriction base="xs:int">
      <xs:minInclusive value="0"/>
   </xs:restriction>
</xs:simpleType>

<!--
   Definition of OPTIMIS EcoEfficiency Constraints.
-->
<xs:complexType name="EcoEfficiencySectionType">
   <xs:sequence>
      <xs:element name="LEEDCertification"
                  type="opt:LEEDCertificationConstraintType"
                  default="NotRequired"/>
```

```
        <xs:element name="BREEAMCertification"
                    type="opt:BREEAMCertificationConstraintType"
                    default="NotRequired"/>
        <xs:element name="EuCoCCompliant" type="xs:boolean" default="false"/>
        <xs:element name="EnergyStarRating" type="opt:EnergyStarRatingType"
                    default="No"/>
        <xs:any namespace="##other" processContents="strict" minOccurs="0"
                maxOccurs="unbounded"/>
      </xs:sequence>
  </xs:complexType>


  <xs:simpleType name="LEEDCertificationConstraintType">
      <xs:restriction base="xs:string">
        <xs:enumeration value="NotRequired"/>
        <xs:enumeration value="Certified"/>
        <xs:enumeration value="Silver"/>
        <xs:enumeration value="Gold"/>
        <xs:enumeration value="Platinum"/>
      </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="BREEAMCertificationConstraintType">
      <xs:restriction base="xs:string">
        <xs:enumeration value="NotRequired"/>
        <xs:enumeration value="Pass"/>
        <xs:enumeration value="Good"/>
        <xs:enumeration value="VeryGood"/>
        <xs:enumeration value="Excellent"/>
        <xs:enumeration value="Outstanding"/>
      </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="EnergyStarRatingType">
      <xs:union>
        <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="No"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
            <xs:restriction base="xs:int">
              <xs:minInclusive value="1"/>
              <xs:maxInclusive value="100"/>
            </xs:restriction>
        </xs:simpleType>
```

```
    </xs:union>
  </xs:simpleType>
  <!--
    Definition of OPTIMIS Cost constraints.

    TODO: Additional input required.
  -->
  <xs:complexType name="CostSectionType">
    <xs:sequence>
      <xs:element name="PricePlan" maxOccurs="unbounded"
                  type="opt:PricePlanType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="PricePlanType">
    <xs:annotation>
      <xs:documentation>
        A PricePlan is a set of charges associated with a network-provisioned
        entity. Alternative sets of fees (i.e. alternative PricePlans) of the
        same service provision may be made available for the consumer to choose
        from, for example to offer the consumer the choice between a flat price
        scheme and a usage-based scheme (a common practice in the
        telecommunication industry). Several PricePlans may exist for the same
        service in order to suit different user profiles and charge them
        appropriately (e.g. heavy- and light-usage users), or as a key price
        customization instrument to individually match diverse service
        valuations. There are three attributes associated with the PricePlan
        term:
        <br/>
        1. currency, as a name string, EString: the currency for all price
        amounts within this PricePlan, e.g. EUR.
        <br/>
        2. planCap, as a float num., EFloat: providing this maximum PricePlan
        value prevents from charging the user a higher total price, regardless
        of the cumulative total price the components and adjustments within this
        PricePlan may eventually amount to. Example: A cap may be used to set an
        upper limit in a strictly usage-based plan.
        <br/>
        3. planFloor, as a float num., EFloat: providing this minimum PricePlan
        value prevents from charging the user a lower total price, regardless of
        the cumulative total price the components and adjustments within this
        PricePlan may eventually amount to. Example: A floor may be used to set
        a lower limit to discounts that may result in an excessively low price.
      </xs:documentation>
    </xs:annotation>
```

```
      <xs:sequence>
         <xs:element name="Scope" type="opt:ScopeArrayType"/>
         <xs:element name="PlanComponents" type="opt:PlanComponentArrayType"/>
      </xs:sequence>
      <xs:attribute name="planCap" type="xs:float"/>
      <xs:attribute name="planFloor" type="xs:float"/>
      <xs:attribute name="currency" type="xs:string"/>
   </xs:complexType>

   <xs:complexType name="PlanComponentArrayType">
      <xs:sequence>
         <xs:element name="PriceComponent" type="opt:PriceComponentType"
maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="PriceComponentType">
      <xs:annotation>
         <xs:documentation>
            PriceComponents are fees included in a PricePlan, which subject to
            conditions (expressed as PriceFences) may contribute to the total amount
            charged. Components within the same plan are summed together in order to
            get the total amount (price of the service). Common examples of
            PriceComponents that may coexist in the same PricePlan are: startup or
            membership charges (to access the service), periodic subscription fees
            (with a certain recurrence - e.g. monthly - as long as committed to by
            the contract), pay-per-unit charges (whose total will be proportional to
            the metered usage), options or feature dependent charges. The final
            value of the component will depend on the active PriceLevel (determined
            by the evaluation of the relative PriceFences) and the PriceAdjustments
            that may apply (e.g. discounts). There are two attributes associated
            with the PriceComponent term:
            <br/>
            1. componentCap, as a float num., EFloat: providing this maximum
            PriceComponent value prevents the component final price from exceeding a
            certain amount, regardless of its levels and the parameters they are
            indexed to. Example: A cap may be used to set an upper limit for a
            component whose levels vary with usage.
            <br/>
            2. componentFloor, as a float num., EFloat: providing this minimum
            PriceComponent value prevents the component final price from falling
            below a certain amount, regardless of its levels and the parameters they
            are indexed to. Example: A floor may be used to set a lower limit for a
            component whose levels vary with usage.
         </xs:documentation>
      </xs:annotation>
```

```
    <xs:sequence>
       <xs:element name="ComponentLevels" type="opt:ComponentLevelArrayType"/>
       <xs:element name="Multiplier" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="componentCap" type="xs:float"/>
    <xs:attribute name="componentFloor" type="xs:float"/>
 </xs:complexType>
 <xs:complexType name="ComponentLevelArrayType">
    <xs:sequence>
       <xs:element name="PriceLevel" type="opt:PriceLevelType"
                   maxOccurs="unbounded"/>
    </xs:sequence>
 </xs:complexType>

 <xs:complexType name="PriceLevelType">
    <xs:annotation>
       <xs:documentation>
          PriceLevel captures amounts charged by a PriceComponent. Since each
          PriceComponent may assume several values depending on the provider's
          price segmentation strategies, it is allowed to contain multiple
          PriceLevels. This allows shaping charged amounts according to customers'
          behavior and aligning usage with capacity or incurred costs (just like
          utilities do by offering different electricity rates for different times
          of day).
       </xs:documentation>
    </xs:annotation>
    <xs:sequence>
       <xs:element name="AbsoluteAmount" type="xs:decimal"/>
       <xs:element name="PriceMetrics" type="opt:PriceMetricArrayType"/>
       <xs:element name="LevelFences" type="opt:LevelFencesArrayType"/>
    </xs:sequence>
 </xs:complexType>

 <xs:complexType name="PriceMetricArrayType">
    <xs:sequence>
       <xs:element name="PriceMetric" type="opt:PriceMetricType"
                   maxOccurs="unbounded"/>
    </xs:sequence>
 </xs:complexType>

 <xs:complexType name="PriceMetricType">
    <xs:annotation>
       <xs:documentation>
          PriceMetric represents the unit of measurement by which the customer is
```

```
            charged for the consumption of the service or bundle. Metrics can be
            abstract/un-typed (e.g. per invocation) or typed (e.g. per MByte). The
            latter are covered by the sub-class TypedPriceMetric. The attributes
            that defines the PriceMetric is the factor, as a float num., EFloat: the
            minimum block of units that is priced, i.e. the step increase the price
            metric may take. It may also be a fraction.
            <br/>
            Examples: - A Gigabyte metric could be expressed equivalently as a
            Megabyte metric with a factor 1024 - A professional service priced with
            hourly rates but charged in 15 minutes increments (factor would be
            0.25).
         </xs:documentation>
      </xs:annotation>
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attribute name="factor" type="xs:float"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>

   <xs:complexType name="LevelFencesArrayType">
      <xs:sequence>
         <xs:element name="PriceFence" type="opt:PriceFenceType"
                     maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="PriceFenceType">
      <xs:annotation>
         <xs:documentation>
            PriceFence represents a conditional expression evaluated to determine if
            a price element (i.e. PricePlan, PriceComponent or PriceLevel) applies.
            Within a PriceFence a certain business entity (represented by the
            businessTerm) is compared to a certain value (or set of values - the
            literals available to account for the different dimensions of the
            service provision process).
         </xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="BusinessTerm" type="xs:string"/>
         <xs:element name="BusinessTermExpression" type="xs:string"/>
         <xs:element name="QuantityLiterals" type="opt:QuantityLiteralsArrayType"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="QuantityLiteralsArrayType">
      <xs:sequence>
         <xs:element name="Quantity" type="opt:QuantityType" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="QuantityType">
      <xs:sequence>
         <xs:element name="Amount" type="xs:decimal"/>
         <xs:element name="TypeReference" type="xs:string"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required"/>
   </xs:complexType>

   <!--

      Definition of the ElasticityArray. The definition of the RuleType is based on the
      Reservoir Elasticity Array. (see schema:
http://schemas.telefonica.com/claudia/ovf)

      TODO: Review required!
   -->
   <xs:element name="ElasticityRule" type="opt:ElasticityRuleType"></xs:element>

   <xs:complexType name="ElasticitySectionType">
      <xs:choice>
         <xs:sequence>
            <xs:element name="SPManagedElasticity" nillable="true"/>
         </xs:sequence>
         <xs:sequence>
            <xs:element name="VariableSet" type="opt:VariableSetType"/>
            <xs:element name="ElasticityRules">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="opt:ElasticityRule" maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:choice>
   </xs:complexType>

   <xs:complexType name="ElasticityRuleType">
      <xs:sequence>
```

```
            <xs:element name="Scope" type="opt:ScopeArrayType"/>
            <xs:element name="Condition" type="opt:ConditionType"/>
            <xs:element name="Effect" type="opt:Effect"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>


    <xs:complexType name="ConditionType">
        <xs:sequence>
            <xs:element name="Expression" type="xs:string"/>
            <xs:element name="AssessmentCriteria" type="opt:AssessmentCriteria"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AssessmentCriteria">
        <xs:sequence>
            <xs:element name="Window" type="xs:duration"/>
            <xs:element name="Frequency" type="xs:int"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="Effect">
        <xs:sequence>
            <xs:element name="Importance" type="xs:int"/>
            <xs:element name="Action" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="Variable" type="opt:VariableType"/>

    <xs:complexType name="VariableSetType">
        <xs:sequence>
            <xs:element ref="opt:Variable" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="VariableType">
        <xs:annotation>
            <xs:documentation>
                Location: the path to a location where the value can be found.
                The type attribute [internal | external] specifies if the variable can be
found internal in the
                manifest itself (location would be an xpath expression) or at some external
            location, e.g URL to REST address of a monitoring system which provides the
current value of
            the variable.
            The metric attribute specifies the type of the value received at location,
eg. int
            The name specifies the variable name which will be used in the rules.
         </xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="Location" type="xs:string"/>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required"/>
      <xs:attribute name="metric" type="xs:string"/>
      <xs:attribute name="type" type="opt:ElasticityLocationTypeEnum" use="required"/>
   </xs:complexType>


   <xs:simpleType name="ElasticityLocationTypeEnum">
      <xs:restriction base="xs:string">
         <xs:enumeration value="internal"/>
         <xs:enumeration value="external"/>
      </xs:restriction>
   </xs:simpleType>


   <xs:complexType name="ScopeArrayType">
      <xs:sequence>
         <xs:element name="ComponentId" type="xs:string" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <xs:simpleType name="PositiveDecimalType">
      <xs:restriction base="xs:decimal">
         <xs:minExclusive value="0"/>
      </xs:restriction>
   </xs:simpleType>
   <!--
      Definition of the OPTIMIS data protection constraints.
   -->
   <xs:complexType name="DataProtectionSectionType">
      <xs:sequence>
         <xs:element name="EligibleCountryList" type="opt:CountryListType"
                     minOccurs="0"/>
         <xs:element name="NonEligibleCountryList" type="opt:CountryListType"
                     minOccurs="0"/>
         <xs:element name="DataProtectionLevel" type="opt:DataProtectionLevelType"
```

```
                        minOccurs="0"/>
            <xs:element name="DataEncryptionLevel" type="opt:EncryptionLevelType"
                        minOccurs="0"/>
            <xs:any namespace="##other" processContents="strict" minOccurs="0"
                    maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:simpleType name="DataProtectionLevelType">
        <xs:annotation>
            <xs:documentation>
                DataProtectionLevel specifies the level of protection that is guaranteed
                by a service provider regarding data management. In general it defines
                to which countries data may be transfered by the provider. Countries are
                divided into countries that have a sufficient level of protection (known
                as Data Protection Area-DPA) and countries that do not meet these
                levels. Transferring sensitive data to the latter is a violation and the
                cloud providers engaged in federations should have the necessary
                framework to prevent this from happening. By law, the Cloud Provider
                does not have the obligation to keep the data in one particular country
                of the DPA. The DataProtectionLevelType specifies whether the data
                included in the service under consideration is sensitive or not. If not,
                there are no limitations to their transfer. If yes, they should be
                restricted to countries that are part of the DPA. The list of the DPA
                countries is the following:

                - all 27 EU Member States - all countries of the European Economic Area
                (Iceland, Liechtenstein, Norway) - Switzerland - Canada - Argentina -
                Guernsey - Isle of Man - US organisations who take part in the US safe
                harbour program - And the state of Israel.

                TODO: Is there a maintained reference list of DPA countries online
                available?
            </xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:enumeration value="DPA"/>
            <xs:enumeration value="None"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="ISO3166Alpha2">
        <xs:annotation>
            <xs:documentation>
                Two-letter (alpha-2) ISO 3166-1 code for one of the 243 countries. These
                codes are subject to change. For valid values refer to
            http://www.iso.org/iso/list-en1-semic-3.txt
         </xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:string">
         <xs:whiteSpace value="collapse"/>
         <xs:pattern value="[A-Z]{2}"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:complexType name="CountryListType">
      <xs:sequence>
         <xs:element name="Country" type="opt:ISO3166Alpha2" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="EncryptionLevelType">
      <xs:choice>
         <xs:sequence>
            <xs:element name="EncryptionAlgoritm"
                        type="opt:EncryptionAlgoritmType"/>
            <xs:element name="EncryptionKeySize" type="xs:int" default="128"
                        minOccurs="0"/>
         </xs:sequence>
         <xs:sequence>
            <xs:element name="CustomEncryptionLevel" type="xs:anyType"/>
         </xs:sequence>
      </xs:choice>
   </xs:complexType>
   <xs:simpleType name="EncryptionAlgoritmType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="NotApplicable"/>
         <xs:enumeration value="AES"/>
         <xs:enumeration value="Twofish"/>
         <xs:enumeration value="AES-Twofish"/>
         <xs:enumeration value="AES-Twofish-Serpent"/>
         <xs:enumeration value="Serpent-AES"/>
         <xs:enumeration value="Serpent-Twofish-AES"/>
         <xs:enumeration value="Twofish-Serpent"/>
      </xs:restriction>
   </xs:simpleType>

   <!--
      Definition of the AffinitySection.

      TODO: Review required!
   -->
```

```
  <xs:element name="AffinityRule" type="opt:AffinityRuleType"/>
  <xs:complexType name="AffinitySectionType">
    <xs:sequence>
      <xs:element ref="opt:AffinityRule" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="AffinityRuleType">
    <xs:sequence>
      <xs:element name="Scope" type="opt:ScopeArrayType"/>
      <xs:element name="AffinityConstraints" type="opt:AffinityConstraintType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="AffinityConstraintType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="High"/>
      <xs:enumeration value="Medium"/>
      <xs:enumeration value="Low"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

8.2 Infrastructure Provider Extensions

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:opt-ip="http://schemas.optimis.eu/optimis/infrastructure"
           xmlns:opt="http://schemas.optimis.eu/optimis/"
           targetNamespace="http://schemas.optimis.eu/optimis/infrastructure" elementFormDefault="qualified"
           attributeFormDefault="qualified">
    <xs:import namespace="http://schemas.optimis.eu/optimis/" schemaLocation="./optimis.xsd"/>

    <xs:element name="InfrastructureProviderExtensions" type="opt-ip:InfrastructureProviderExtensionType"
                nillable="true"/>
    <xs:element name="IncarnatedServiceComponents" type="opt-ip:IncarnatedServiceComponentsType"/>
    <xs:element name="IncarnatedServiceComponent" type="opt:ServiceComponentType"/>

    <xs:element name="IncarnatedVirtualMachineComponents" type="opt-ip:IncarnatedVirtualMachineComponentsType"
                substitutionGroup="opt-ip:IncarnatedServiceComponents"/>
    <xs:element name="IncarnatedVirtualMachineComponent" type="opt-ip:IncarnatedVirtualMachineComponentType"
                substitutionGroup="opt-ip:IncarnatedServiceComponent"/>

    <xs:complexType name="InfrastructureProviderExtensionType">
        <xs:annotation>
            <xs:documentation>
                Provisioning of extensions for an Infrastructure Provider.
            </xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element ref="opt-ip:IncarnatedServiceComponents" minOccurs="0"/>
            <xs:any namespace="##other" processContents="strict" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="IncarnatedServiceComponentsType" abstract="true">
        <xs:sequence>
            <xs:element ref="opt-ip:IncarnatedServiceComponent" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="IncarnatedVirtualMachineComponentsType">
        <xs:complexContent>
            <xs:restriction base="opt-ip:IncarnatedServiceComponentsType">
                <xs:sequence>
                    <xs:element ref="opt-ip:IncarnatedVirtualMachineComponent" minOccurs="1" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="IncarnatedVirtualMachineComponentType">
        <xs:complexContent>
            <xs:extension base="opt:ServiceComponentType">
                <xs:sequence>
                    <xs:element ref="opt:OVFDefinition" maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

8.3 Service Provider Extensions

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:opt-sp="http://schemas.optimis.eu/optimis/service"
           targetNamespace="http://schemas.optimis.eu/optimis/service"
elementFormDefault="qualified"
           attributeFormDefault="qualified">

   <xs:import namespace="http://schemas.dmtf.org/ovf/envelope/1"
       schemaLocation="http://schemas.dmtf.org/ovf/envelope/1/dsp8023_1.1.0.xsd"/>
   <xs:import namespace=http://schemas.optimis.eu/optimis/
       schemaLocation="./optimis.xsd"/>

   <xs:element name="ServiceProviderExtensions"
        type="opt-sp:ServiceProviderExtensionType"/>
   <xs:element name="VirtualMachineComponentConfiguration"
        type="opt-sp:VirtualMachineComponentConfigurationType"/>

   <xs:complexType name="ServiceProviderExtensionType">
      <xs:annotation>
         <xs:documentation>
            Provisioning of extensions for a Service Provider.
         </xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element ref="opt-sp:VirtualMachineComponentConfiguration"
maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="VirtualMachineComponentConfigurationType">
      <xs:sequence>
         <xs:element name="SoftwareDependencies"
              type="opt-sp:SoftwareDependencyArrayType"/>
         <xs:element name="SecurityKeyType"
              type="opt-sp:SecurityKeyType" default="None"/>
         <!-- This can be used to add any desired configuration by
              using key, value, type -->
         <xs:element name="ComponentProperties"
               type="opt-sp:ComponentPropertyArrayType" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="componentId" type="xs:string"/>
   </xs:complexType>


   <xs:complexType name="SoftwareDependencyArrayType">
      <xs:sequence>
         <xs:element name="Dependency" minOccurs="0" maxOccurs="unbounded">
```

```
            <xs:complexType>
               <xs:all>
                  <xs:element name="groupId" type="xs:string"/>
                  <xs:element name="artifactId" type="xs:string"/>
                  <xs:element name="version" type="xs:string"/>
               </xs:all>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>

   <xs:simpleType name="SecurityKeyType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="SSH"/>
         <xs:enumeration value="VPN"/>
         <xs:enumeration value="None"/>
      </xs:restriction>
   </xs:simpleType>

   <xs:complexType name="ComponentPropertyArrayType">
      <xs:sequence>
         <xs:element name="ComponentProperty"
            type="opt-sp:ComponentPropertyType" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="ComponentPropertyType">
      <xs:sequence>
         <xs:element name="Name" type="xs:string"/>
         <xs:element name="Value" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
</xs:schema>
```