ERICSSON

Atlas OVF to HOT Mapping

INTERWORK DESCRIPTION

6/155 19-CRA 119 1873/5 Uen B

Copyright

© Ericsson AB 2016. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

Trademark List

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.

Contents

1 Introduction

The Open Virtualization Format (OVF) is an industry used distribution format for virtual machines (VMs) and collections of VMs with related support artifacts such as network. OVF provides a platform-independent, efficient, extensible, and open packaging and distribution format that meets these needs.

Heat Orchestration Template (HOT) is the template format used for orchestration of stacks, VMs, and related support artifacts, using the OpenStack Heat service.

This document describes a mechanism for the translation of OVF descriptors into HOT templates.

1.1 Scope

Although this document provides a mapping for each OVF section, the scope is limited to the format supported in HOT itself. If there is no scope for a section to be migrated to HOT, then no mapping is provided. If possible, an alternative must be proposed on how the section (or, at least, the functionality of that section) can be handled.

This document discusses how OVF packages are deployed. The creation process of OVF packages, with the whole software structure in place, is out of the scope of this document. Any OVF package that confers to the OVF standards and specifications can be deployed by this functionality.

2 Open Virtualization Format

The OVF specification describes an open, secure, efficient, and extensible format for packaging and distribution of software to be run in virtual systems.

The OVF package enables the authoring of portable virtual systems and the transport of virtual systems between virtualization platforms.

An OVF package must include the following file:

- One OVF descriptor, with the extension `ovf`

Additionally, an OVF package may include the following files:

- Zero or one OVF manifest, with the extension `mf`
- Zero or one OVF certificate, with the extension `cert`
- Zero or more disk image files
- Zero or more additional resource files, such as ISO images

An OVF package can be stored either as a single compressed file, `OVA`, or a set of files. Both modes must be supported.

The manifest and certificate files, when present, must not be included in the `References` section of the OVF descriptor. This ensures that the OVF descriptor content does not depend on whether the OVF package has a manifest or is signed. Furthermore, the decision to add a manifest or certificate to a package can be deferred to a later stage.

2.1 OVF Descriptor

The OVF descriptor contains metadata about the OVF package. This is an extensible XML document for encoding information, such as product details, virtual hardware requirements, and licensing. Elements used as part of the OVF descriptor are discussed in brief below. For more information, refer to Open Virtualization Format Specification – version 2.1, Reference [1].

2.1.1 Envelope Element

The `Envelope` element describes all metadata for the virtual systems, including virtual hardware, as well as the structure of the OVF package itself.

2.1.2 Reference Element

The `Reference` element contains the references of the files used for virtual system definitions.

2.1.3 Content Element

The `Content` element is abstract and cannot be used directly. In the OVF schema, the `VirtualSystem` and `VirtualSystemCollection` elements are part of a substitution group with the `Content` element as the head of the substitution group. The OVF descriptor must have one or more `Content` elements.

The `VirtualSystem` element describes a single virtual system and is a container of section elements. These section elements describe virtual hardware, resources, and product information.

The `VirtualSystemCollection` element is a container of zero or more `VirtualSystem` or `VirtualSystemCollection` elements. Thus, arbitrary complex configurations can be described. The section elements at the `VirtualSystemCollection` level describe appliance information, properties, and resource requirements.

2.1.4 Virtual Hardware Description

The `VirtualHardwareSection` element is used to describe the virtual hardware used by the virtual system. The virtual hardware devices listed in the `VirtualHardwareSection` element must be realized. A `VirtualSystem` element must have a `VirtualHardwareSection` direct child element. `VirtualHardwareSection` must not be a direct child element of a `VirtualSystemCollection` element or of an `Envelope` element. One or more `VirtualHardwareSection` elements can occur within a `VirtualSystem` element.

Table 1 shows the core metadata sections defined in the OVF namespace.

Table 1 Core Metadata Sections in OVF Namespace

Section Element	Parent Element	Multiplicity
`DiskSection` Describes meta information about all virtual disks in the package.	`Envelope`	Zero or one
`NetworkSection` Describes logical networks used in the package.	`Envelope`	Zero or one

Section Element	Parent Element	Multiplicity
`ecs:VirtualRouterSection` This Ericsson extension section defines virtual routers.	`Envelope`	Zero or one
`ecs:VirtualNetworkRouterInterfaceSection` This Ericsson extension section defines virtual router interfaces.	`Envelope`	Zero or one
`ResourceAllocationSection` Specifies reservations, limits, and shares on a given resource, such as memory or CPU for a virtual system collection.	`VirtualSystemCollection`	Zero or one

`AnnotationSection`
 	
Specifies a free-form annotation on an entity.
	`VirtualSystem`
	
	`VirtualSystemCollection`
		Zero or one

`ProductSection`
 	
Specifies product-information for a package, such as product name and version, along with a set of properties that can be configured.
	`VirtualSystem`
	
	`VirtualSystemCollection`
		Zero or more

`EulaSection`
 	
Specifies a license agreement for the software in the package.
	`VirtualSystem`
	
	`VirtualSystemCollection`
		Zero or more

`StartupSection` Specifies how a virtual system collection is powered on.	`VirtualSystemCollection`	Zero or one
`DeploymentOptionSection` Specifies a discrete set of intended resource requirements.	`Envelope`	Zero or one
`OperatingSystemSection` Specifies the guest software installed on a virtual system.	`VirtualSystem`	Zero or one
`InstallSection` Specifies that the virtual system needs to be initially booted, in order that the software can be installed and configured.	`VirtualSystem`	Zero or one

Section Element	Parent Element	Multiplicity
`EnvironmentFilesSection` Specifies additional files from an OVF package to be included in the OVF environment.	`VirtualSystem`	Zero or one
`BootDeviceSection` Specifies boot device order to be used by a virtual system.	`VirtualSystem`	Zero or more
`SharedDiskSection` Specifies virtual disks shared by more than one `VirtualSystem`s at runtime.	`Envelope`	Zero or more
`ScaleOutSection` Specifies that a `VirtualSystemCollection` contains a set of children that are homogeneous with respect to a prototype.	`VirtualSystemCollection`	Zero or more
`PlacementGroupSection` Specifies a placement policy for a group of `VirtualSystem`s or `VirtualSystemCollection`s.	`Envelope`	Zero or more

`PlacementSection`
 	
Specifies membership of a particular placement policy group.
	`VirtualSystem`
	
	`VirtualSystemCollection`
		Zero or more

`EncryptionSection` Specifies the encryption scheme for encrypting parts of an OVF descriptor or file to which it refers.	`Envelope`	Zero or more

3 Heat Orchestration Template

HOT is a template format that replaces the Heat `CloudFormation` compatible format (CFN) as the native format supported by Heat.

A HOT template is an orchestration document that details everything that is needed to carry out an orchestration. It is expressed as YAML. HOT templates may be shared among multiple tenants of different cloud service providers. They can contain all of the vendor independent specifications for launching a particular service or application.

HOT is considered reliable, supported, and standardized as of the Icehouse (April 2014) release. The Heat core team may make improvements to the standard, which is likely to be backward compatible. The template format is also versioned. Since the Juno release, Heat supports multiple different versions of the HOT specification.

HOT templates are defined in YAML and the structure is outlined as follows:

```
heat_template_version: 2013-05-23
description:
  # a description of the template
parameter_groups:
  # a declaration of input parameter groups and order
parameters:
  # declaration of input parameters
resources:
  # declaration of template resources
outputs:
  # declaration of output parameters
```

The detailed information about the HOT template is as follows:

heat_template_version

This key with value 2013-05-23 (or a later date) indicates that the YAML document is a HOT template of the specified version.

description

This optional key allows for giving a description of the template, or the workload that can be deployed using the template.

parameter_groups

This section allows for specifying how the input parameters can be grouped and the order to provide the parameters in. This section is optional and can be omitted when necessary.

parameters	This section allows for specifying input parameters that have to be provided when instantiating the template. The section is optional and can be omitted when no input is required.
resources	This section contains the declaration of the single resources of the template. This section with at least one resource must be defined in any HOT template, or the template would not function when being instantiated.
outputs	This section allows for specifying output parameters available to users once the template has been instantiated. This section is optional and can be omitted when no output values are required.

4 OVF in Heat for Orchestration

Since Heat is responsible for orchestration, ensure that OVF packages can be orchestrated using Heat.

As a workaround for the limitations of Heat, a pre-deployment module is used, which covers the functionality of OVF deployment specifications other than the orchestration itself.

The module must:

- Validate the OVF package
- Unpack the OVF package
- Upload images to Glance
- Convert descriptor from OVF/TOSCA to HOT
- Store HOT template in catalog database

The architecture of the pre-deployment module is shown in Figure 1.

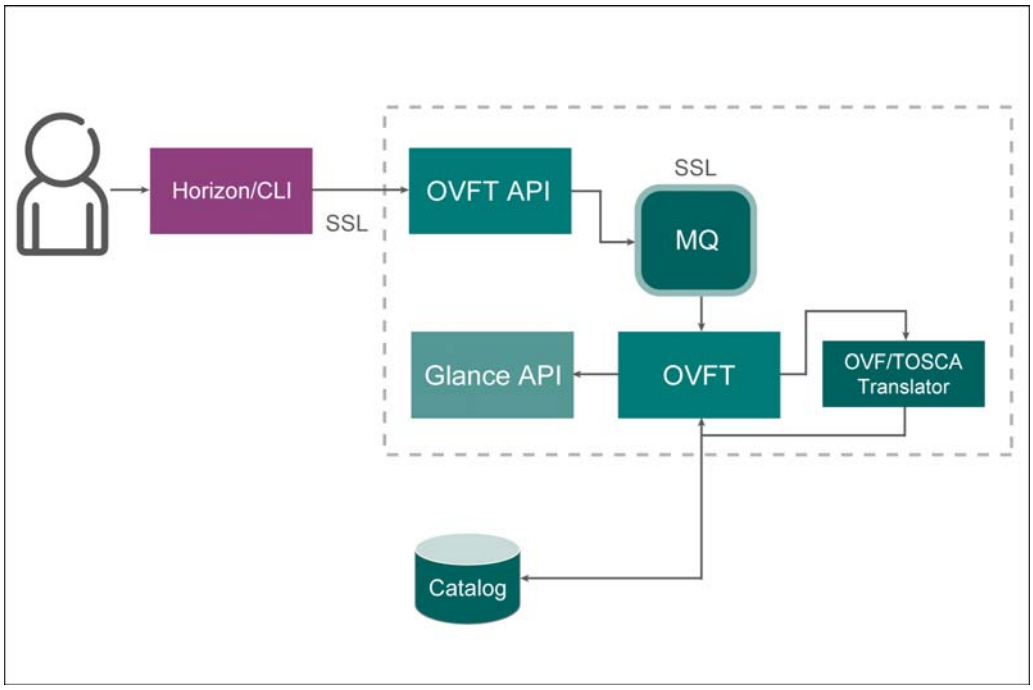

Figure 1 OVFT Architecture, Pre-Deployment Module

4.1 OVF-HOT Mapping

The most important aspect of achieving the deployment of an OVF package via Heat orchestration is to have the virtual system and network definitions in OVF format converted to HOT format.

The conversion must ensure that the configuration launched satisfies OVF specifications.

4.1.1 DiskSection Interpretation

The `DiskSection` element describes meta information about the virtual disks in the OVF package. The virtual disks and associated metadata are described outside of the `VirtualHardwareSection` element to facilitate sharing between the virtual systems within an OVF package.

An example of the `DiskSection` element is as follows:

```
<DiskSection>
   <Info>Virtual disk information</Info>
   <Disk ovf:capacity="200" ovf:capacityAllocationUnits="byte * 2^30"
    ovf:diskId="vmdisk1" ovf:fileRef="file1"
    ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#streamOptimized" />
   <Disk ovf:capacity="50" ovf:capacityAllocationUnits="byte * 2^30"
    ecs:volumeType="SATA" ovf:diskId="bsv1"  ovf:fileRef="file2"
    ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#streamOptimized" />
</DiskSection>
```

The `DiskSection` element shows the capacity of the disk represented by the value in `ovf:capacity`. This value must be used during the flavor selection or creation.

`DiskSection` →`ovf:capacity` →`Ericsson::Nova::Flavor` (property: `disk`)

The value in `ecs:volumeType` defines the volume type for block storage volume. This value can be `SATA` or `SSD`.

The defined volume type is not mapped to the property volume type of `OS::Cinder::Volume`, as CEE does not support different volume types. A Cinder volume resource is created in the HOT template for `ecs:volumeTyp` equal to `SATA` or `SSD`, and `ovf:capacity` is used to define the size of the Cinder volume.

4.1.2 NetworkSection Interpretation

The `NetworkSection` element must list all logical networks used in the OVF package.

`NetworkSection` is only valid as a direct child element of the `Envelope` element. A `Network` element is a child element of `NetworkSection`.

All networks referred to from `Connection` elements in all `VirtualHardware Section` elements must be defined in `NetworkSection`.

Each logical network may contain a set of networking attributes that must be applied when mapping the logical network at deployment time to a physical or virtual network. Networking attributes are specified by zero or more instances of the `NetworkPortProfile` child element or the `NetworkPortProfileURI` child element of the `Network` element.

An example of `NetworkSection` is as follows:

```
<NetworkSection>
   <Info>The list of logical networks</Info>
   <Network ovf:name="network"
       ecs:category="layer3"
      ecs:enableDHCP="true"
      ecs:IPAddressRange="10.33.168.64/27"
       ecs:useEcmSegmentationId="false"
       ecs:segmentationId="3950">

    <Description>The network</Description>
   <NetworkPortProfile>
      <Item>
         <epasd:AllowedToTransmitVLANs>100<epasd:AllowedToTransmitVLANs>
         <epasd:AllowedToTransmitVLANs>200<epasd:AllowedToTransmitVLANs>
         <epasd:AllowedToReceiveVLANs>100<epasd:AllowedToReceiveVLANs>
         <epasd:AllowedToReceiveVLANs>200<epasd:AllowedToReceiveVLANs>
      </Item>
   </NetworkPortProfile>
   </Network>
</NetworkSection>
```

`NetworkSection` specifications can be mapped to OS::Neutron::Net, OS::Neutron::ProviderNet, OS::Neutron::Subnet, and Ericsson::Neutron::Port and can use the network names while creating the VMs (in OS::Nova::Server).

4.1.3 VirtualRouterSection Interpretation

This extension section defines Ericsson Cloud Manager (ECM) virtual routers (VRFs) in the OVF file. This top-level element contains a list of new virtual router elements.

An example of `VirtualRouterSection` is as follows:

```
<ecs:VirtualRouterSection>
  <Info>Virtual Router Section</Info>
  <ecs:VirtualRouter
    ecs:id="vfr1"
    ecs:name="VRF_OVF1"/>
 </ecs:VirtualRouterSection>
```

The `VirtualRouterSection` specifications can be mapped to OS::Neutron::Router.

4.1.4 VirtualNetworkRouterInterfaceSection Interpretation

This extension section defines ECM Virtual Router Network Interface Cards (VRF VNICs) in the OVF file. This top-level element contains a list of new virtual router elements.

An example of `VirtualNetworkRouterInterfaceSection` is as follows:

```
<ecs:VirtualNetworkRouterInterfaceSection>
<Info>VN Router Interface Section</Info>
<ecs:VirtualNetworkRouterInterface
     ecs:id="vfr_vnic1"
     ecs:name="VRF_VNIC_OVF1"
     ecs:category="internalVrfInterface"
     ecs:virtualRouterRef="vfr1"
     ecs:networkRef="Network_OVF"/>
</ecs:VirtualNetworkRouterInterfaceSection>
```

The `VirtualNetworkRouterInterfaceSection` specifications can be mapped to OS::Neutron::RouterInterface and Ericsson::Neutron::Port.

4.1.5 ResourceAllocationSection Interpretation

The `ResourceAllocationSection` element describes all resource allocation requirements of a `VirtualSystemCollection` entity and applies only to the direct child `VirtualSystem` elements that do not contain a `VirtualHardwareSection` element. It does not apply to child `VirtualSystemCollection` elements.

An example of `ResourceAllocationSection` is as follows:

```
<ResourceAllocationSection>
<Info>Defines reservations for CPU and memory for the collection of VSs</Info>
<Item>
<rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
<rasd:ElementName>300 MB reservation</rasd:ElementName>
<rasd:InstanceID>0</rasd:InstanceID>
<rasd:Reservation>300</rasd:Reservation>
<rasd:ResourceType>4</rasd:ResourceType>
</Item>
<EthernetPortItem>
<epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
<epasd:Connection>VS Network</epasd:Connection>
<epasd:Description>Virtual NIC</epasd:Description>
<epasd:ElementName>Ethernet Port 1</epasd:ElementName>
<epasd:InstanceID>3</epasd:InstanceID>
<epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
<epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
<epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
</EthernetPortItem>
<StorageItem>
<sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
<sasd:Description>Virtual Disk</sasd:Description>
<sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
<sasd:InstanceID>4</sasd:InstanceID>
<sasd:Reservation>100</sasd:Reservation>
<sasd:ResourceType>31</sasd:ResourceType>
<sasd:VirtualQuantity>1</sasd:VirtualQuantity>
</StorageItem>
</ResourceAllocationSection>
```

ResourceAllocationSection must be treated the same way as VirtualHardwareSection since a VirtualSystemSection that does not contain VirtualHardwareSection must refer to ResourceAllocationSection for VirtualHardware details. Currently ResourceAllocationSection is not supported.

4.1.6 AnnotationSection Interpretation

The AnnotationSection element is a user-defined annotation on an entity. If present, the AnnotationSection element may be displayed during deployment of the OVF package.

An example of AnnotationSection is as follows:

```
<AnnotationSection>
<Info>An annotation on this service. It can be ignored</Info>
<Annotation>Contact customer support if you have any problems</Annotation>
</AnnotationSection>
```

The section provides a remark for the VMs deployed. Heat does not have a resource type yet for including the properties of this section.

In the overview tab for stacks each launched instance has a description section. This information can be passed to that section. The parameter in consideration is still being analyzed. Currently AnnotationSection is not supported.

4.1.7 ProductSection Interpretation

The ProductSection element specifies product information for an appliance, such as product name, version, and vendor. Typically it corresponds to a particular software product that is installed.

The Product element is optional and specifies the name of the product.

The Vendor element is optional and specifies the name of the product vendor.

The Version element is optional and specifies the product version in short form.

The FullVersion element is optional and describes the product version in long form.

The ProductUrl element is optional and specifies a URL that must resolve to a human-readable description of the product.

The VendorUrl element is optional and specifies a URL that must resolve to a human-readable description of the vendor.

The AppUrl element is optional and specifies a URL resolving to the deployed product instance.

The `Icon` element is optional and specifies display icons for the product.

The `Property` elements specify customization parameters and are relevant to the appliances that need to be customized during deployment with specific settings such as network identity, the IP addresses of DNS servers, gateways, and others.

The ID and the value of the `Property` elements are exposed to the guest software using the OVF environment file.

An example of `ProductSection` is as follows:

```
<ProductSection ovf:class="com.ericsson.cba" ovf:instance="1">
   <Info>Information about the installed software</Info>
   <Product>Ericsson CSCF</Product>
   <Vendor>Ericsson AB</Vendor>
   <Version>14A</Version>
   <FullVersion>14A</FullVersion>
   <Property ovf:key="application.type" ovf:type="string" ovf:value="CSCF">
   <Label>APPLICATION_TYPE</Label>
     <Description>APPLICATION_TYPE</Description>
   </Property>
   <Category>Domain configuration</Category>
   <Property ovf:key="domain.name" ovf:type="string" ovf:userConfigurable="true" ovf:value="ims.ericloud">
     <Label>DOMAINNAME</Label>
     <Description>DOMAINNAME</Description>
   </Property>
   <Category>G3 Configuration</Category>
   <Property ovf:key="g3.subnetwork" ovf:type="string" ovf:userConfigurable="true" ovf:value="Germany">
     <Label>G3_SUBNETWORK</Label>
     <Description>G3_SUBNETWORK</Description>
   </Property>
   <Category>IO network</Category>
   <Property ovf:key="io.oam.subnet" ovf:type="string" ovf:userConfigurable="true"⇒
ovf:value="10.63.69.96/27">
     <Label>IO_OAM_SUBNET</Label>
     <Description>IO_OAM_SUBNET</Description>
   </Property>
   <Property ovf:key="io.oam.gateway" ovf:type="string" ovf:userConfigurable="true" ovf:value="10.63.69.97">
     <Label>IO_OAM_GATEWAY</Label>
     <Description>IO_OAM_GATEWAY</Description>
   </Property>
   <Property ovf:key="io.oam.sc1" ovf:type="string" ovf:userConfigurable="true" ovf:value="10.63.69.100">
     <Label>IO_OAM_SC1_IP</Label>
     <Description>IO_OAM_SC1_IP</Description>
   </Property>
   <Property ovf:key="io.oam.sc2" ovf:type="string" ovf:userConfigurable="true" ovf:value="10.63.69.101">
     <Label>IO_OAM_SC2_IP</Label>
     <Description>IO_OAM_SC2_IP</Description>
   </Property>
   <Category>OAM VIP addresses</Category>
   <Property ovf:key="oam.vip" ovf:type="string" ovf:userConfigurable="true" ovf:value="10.63.69.164">
     <Label>OAM_VIP</Label>
     <Description>OAM_VIP</Description>
   </Property>
   <Category>Time configuration</Category>
   <Property ovf:key="time.server" ovf:type="string" ovf:userConfigurable="true" ovf:value="10.63.224.254">
     <Label>TIME_SERVER</Label>
     <Description>TIME_SERVER</Description>
   </Property>
   <Property ovf:key="time.zone.region" ovf:type="string" ovf:userConfigurable="true" ovf:value="Europe">
     <Label>TIME_ZONE_REGION</Label>
     <Description>TIME_ZONE_REGION</Description>
   </Property>
   <Property ovf:key="time.zone.city" ovf:type="string" ovf:userConfigurable="true" ovf:value="Berlin">
     <Label>TIME_ZONE_CITY</Label>
     <Description>TIME_ZONE_CITY</Description>
   </Property>
   <Category>Traffic VIP addresses</Category>
   <Property ovf:key="pcscf.traffic.vip" ovf:type="string" ovf:userConfigurable="true"⇒
ovf:value="10.63.69.180">
     <Label>PCSCF_VIP</Label>
     <Description>PCSCF_VIP</Description>
   </Property>
   <Property ovf:key="icscf.traffic.vip" ovf:type="string" ovf:userConfigurable="true" ovf:value="10.63.69.181">
     <Label>ICSCF_VIP</Label>
     <Description>ICSCF_VIP</Description>
   </Property>
   <Property ovf:key="scscf.traffic.vip" ovf:type="string" ovf:userConfigurable="true" ovf:value="10.63.69.182">
     <Label>SCSCF_VIP</Label>
     <Description>ECSCF_VIP</Description>
   </Property>
   <Property ovf:key="ecscf.traffic.vip" ovf:type="string" ovf:userConfigurable="true" ovf:value="10.63.69.183">
     <Label>ECSCF_VIP</Label>
     <Description>ECSCF_VIP</Description>
   </Property>
 </ProductSection>
 <ProductSection>
   <Info>Information about the installed software</Info>
   <Product>CSCF</Product>
   <Vendor>Ericsson</Vendor>
   <Version>13A</Version>
   <FullVersion>CXP9999999_8-R111111</FullVersion>
 </ProductSection>
```

The `ProductSection` details which will be part of the OVF environment file, used after instance boot-up, can be provided in HOT format by using the `personality:Map` property of the `OS::Nova::Server`.

The details of ProductSection are also placed into the metadata:Map property of the OS::Nova::Server.

OVF Env file → OS::Nova::Server with the following properties:

Personality:Map A map of files to create or overwrite on the server upon boot. Keys are file names and values are the file contents.

Metadata:Map Arbitrary key/value metadata to store for the server. Both keys and values must be 255 characters or less. Non-string values are serialized to JSON (and the serialized string must be 255 characters or less).

4.1.8 EulaSection Interpretation

The EulaSection element contains the legal terms for using its parent Content element. Multiple EulaSections may be present in an OVF. The EulaSection is a valid section for a VirtualSystem and a VirtualSystemCollection entity.

An example of EulaSection is as follows:

```
<EulaSection>
<Info>Licensing agreement</Info>
<License>
This is the license agreement for the Package Distribution.
Please do accept the terms and conditions before proceeding with deployment
</License>
</EulaSection>
```

The current understanding is that this License Agreement needs to be prompted for acceptance by the user before proceeding with the launch of VMs. Currently EulaSection is not supported.

4.1.9 StartupSection Interpretation

The StartupSection element specifies how a collection of virtual systems identified by a VirtualSystemCollection element is powered on and off. The StartupSection element must not be part of a VirtualSystem element.

An example of StartupSection is as follows:

```
<StartupSection>
      <Info>Startup order</Info>
   <Item ovf:id="SC1" ovf:order="1" ovf:startAction="powerOn"
 ovf:startDelay="0" ovf:stopAction="powerOff" ovf:stopDelay="120" />
   <Item ovf:id="SC2" ovf:order="2" ovf:startAction="powerOn"
 ovf:startDelay="60" ovf:stopAction="powerOff" ovf:stopDelay="120" />
   <Item ovf:id="PL3" ovf:order="3" ovf:startAction="powerOn"
 ovf:startDelay="10" ovf:stopAction="powerOff" ovf:stopDelay="120" />
   <Item ovf:id="PL4" ovf:order="4" ovf:startAction="powerOn"
 ovf:startDelay="10" ovf:stopAction="powerOff" ovf:stopDelay="120" />
</StartupSection>
```

The HOT template provides the following resource, which induces a delay between VM launches (one VM waits for the previous one to be booted): `Ericsson::Heat::DelayPolicy`

4.1.10 DeploymentSection Interpretation

The `DeploymentOptionSection` element specifies a discrete set of intended resource configurations. The author of an OVF package can include sizing metadata for different configurations. The deployment selects one of the configurations, for example, by prompting the user. The selected configuration must be available in the OVF environment file.

Configurations may be used to control resources for virtual hardware and for virtual system collections. The `Item`, `EthernetPortItem`, and `StorageItem` elements in the `VirtualHardwareSection` elements describe resources for `VirtualSystem` entities, while the `Item`, `EthernetPortItem`, and `StorageItem` elements in the `ResourceAllocationSection` elements describe resources for virtual system collections.

An example of `DeploymentSection` is as follows:

```
<DeploymentOptionSection>
<Configuration ovf:id="minimal">
<Label>Minimal</Label>
<Description>Some description</Description>
</Configuration>
<Configuration ovf:id="normal" ovf:default="true">
<Label>Typical</Label>
<Description>Some description</Description>
</Configuration>
<Configuration ovf:id="big">
<Label>Typical</Label>
<Description>Some description</Description>
</Configuration>
</DeploymentOptionSection>


<VirtualHardwareSection>
<Info>...</Info>
<Item>
<rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
<rasd:ElementName>512 MB memory size and 256 MB reservation</rasd:ElementName>
<rasd:InstanceID>0</rasd:InstanceID>
<rasd:Reservation>256</rasd:Reservation>
<rasd:ResourceType>4</rasd:ResourceType>
<rasd:VirtualQuantity>512</rasd:VirtualQuantity>
</Item>

<Item ovf:configuration="big">
<rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
<rasd:ElementName>1024 MB memory size and 512 MB reservation</rasd:ElementName>
<rasd:InstanceID>0</rasd:InstanceID>
<rasd:Reservation>512</rasd:Reservation>
<rasd:ResourceType>4</rasd:ResourceType>
<rasd:VirtualQuantity>1024</rasd:VirtualQuantity>
</Item>
</VirtualHardwareSection>
```

If the configuration section exists, the vCPU and RAM user is prompted to select the desired type of configuration and accordingly the HOT format is built.

4.1.11 OperatingSystemSection Interpretation

The `OperatingSystemSection` element specifies the operating system installed on a virtual system. The `OperatingSystemSection` element is a valid section only for a `VirtualSystem` element.

An example of `OperatingSystemSection` is as follows:

```
<OperatingSystemSection>
   <Info>The kind of installed guest operating system⇒
</Info>
   <Description>Payload Processor</Description>
</OperatingSystemSection>
```

There is no resource type in HOT that supports this section. The OS type could be added as image metadata before uploading the image on Glance. Currently `OperatingSystemSection` is not supported.

4.1.12 InstallSection Interpretation

The `InstallSection` element, if specified, indicates that the virtual system needs to be booted once in order to install and configure, or only configure the guest software. The guest software accesses the OVF environment during that boot and shuts down after having completed the installation and configuration, or only configuration of the software, powering off the guest.

An example of `InstallSection` is as follows:

```
<InstallSection ovf:initialBootStopDelay="300">
<Info>Specifies that the virtual system needs to be booted once after
 having created the guest software in order to install and/or
 configure the software </Info>
</InstallSection>
```

`InstallSection` does not have a mapping in HOT. This can be handled in the pre-deployment phase.

The mechanism here is to execute or transport the Environment file for the VM being considered and after the Delay, send the Nova command to shut the VM down. Currently `InstallSection` is not supported.

4.1.13 EnvironmentFilesSection Interpretation

The `EnvironmentFilesSection` element enables the OVF package to specify Additional Environment Files (AEFs) besides the virtual disks. These AEFs enable increased flexibility in image customization outside of virtual disk capture, allowing an OVF package to provide customized solutions by combining existing virtual disks without modifying them.

AEF contents are neither generated nor validated by the deployment function. The AEFs are included in the transport media generated by the deployment function. The AEFs are conveyed to the guest software by using the indicated transport media type. The AEFs and the OVF environment files use the same

transport media and transport media type. The `EnvironmentFilesSection` must contain a `File` element with the attributes `ovf:fileRef` and `ovf:path` for each AEF provided to the guest software.

An example of `EnvironmentFilesSection` is as follows:

```
<Envelope>
<References>
...
<File ovf:id="config" ovf:href="config.xml" ovf:size="4332"/>
</References>
...
<VirtualSystem ovf:id="...">
...
<ovf:EnvironmentFilesSection ovf:required="false" ovf:transport="iso">
<Info>Config files to be included in OVF environment</Info>
<ovf:File ovf:fileRef="config" ovf:path="setup/cfg.xml"/>
</ovf:EnvironmentFilesSection>
...
</VirtualSystem>
...
</Envelope>
```

The OVF environment file, used after instance boot-up, can be provided in HOT format by using the `personality:Map` property of the `OS::Nova::Server`.

`OVF Env` file →`OS::Nova::Server` (property – `personality:Map`)

`Personality:Map`

A map of files to create/overwrite on the server upon boot. Keys are file names and values are the file contents.

The file with `ovf:path="cloud-init-data"` or `user-data.txt` is treated as the designated User-Data (AEF) file, which can be provided in HOT format by using the `user_data` property of the `OS::Nova::Server` with `user_data_format="RAW"`.

Note: There is a limitation posed by OpenStack for mounting an ISO image towards a VM for transporting OEFs. This enhancement needs to be proposed in OpenStack.

4.1.14 BootDeviceSection Interpretation

Individual virtual systems use the default device boot order provided by the virtual BIOS of the virtualization platform. The `BootDeviceSection` allows the OVF package author to specify particular boot configurations and boot order settings. This enables booting from non-default devices, such as a NIC using Preboot Execution Environment (PXE), a USB device, or a secondary disk. Moreover, there could be multiple boot configurations with different boot orders. For example, a virtual disk may need to be patched before it is bootable and a patch ISO image could be included in the OVF package.

An example of `BootDeviceSection` is as follows:

```
<ovf:BootDeviceSection>
 <Info>Boot device order specification</Info>
 <bootc:CIM_BootConfigSetting>
  <bootc:Caption>Diskless</bootc:Caption>
  <bootc:Description>Boot Sequence for diskless node</bootc:Description>
  <boots:CIM_BootSourceSetting>
   <boots:Caption>Boot from internal-0</boots:Caption>
   <boots:InstanceID>8</boots:InstanceID>
  </boots:CIM_BootSourceSetting>
  <boots:CIM_BootSourceSetting>
   <boots:Caption>Boot from internal-1</boots:Caption>
   <boots:InstanceID>9</boots:InstanceID>
  </boots:CIM_BootSourceSetting>
 </bootc:CIM_BootConfigSetting>
</ovf:BootDeviceSection>
```

The scope of OpenStack for selecting a boot device is limited.

CD-ROM is not supported yet.

For a diskless PXE booting, the image must be pre-created depending on the application needs. Currently `BootDeviceSection` is not supported.

4.1.15 SharedDiskSection Interpretation

Certain applications, such as clustered databases, rely on multiple virtual systems sharing the same virtual disk at runtime. `SharedDiskSection` allows the OVF package to specify `Disk` elements shared by more than one virtual system at runtime. These virtual disks may be backed by an external File reference, or may be empty virtual disks without backing. It is recommended that the guest software use cluster-aware file system technology to be able to handle concurrent access.

An example of `SharedDiskSection` is as follows:

```
<ovf:SharedDiskSection>
 <Info>Describes the set of virtual disks shared between VSs</Info>
 <ovf:SharedDisk ovf:diskId="datadisk" ovf:fileRef="data"
  ovf:capacity="8589934592" ovf:populatedSize="3549324972"
 ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#sparse"/>
 <ovf:SharedDisk ovf:diskId="transientdisk" ovf:capacity="536870912"/>
</ovf:SharedDiskSection>
```

Currently `SharedDiskSection` is not supported.

4.1.16 ScaleOutSection Interpretation

The number of virtual systems or collections of virtual systems contained in an OVF package is fixed and determined by the structure inside the `Envelope` element. The `ScaleOutSection` element allows a `VirtualSystemCollection` element to contain a set of children that are homogeneous with respect to a prototypical `VirtualSystem` or `VirtualSystemCollection` element. The `ScaleOutSection` element must cause the deployment function to replicate the prototype a number of times, thus allowing the number of instantiated virtual systems to be configured dynamically at deployment time.

This mechanism enables scaling of virtual system instances at deployment time. Scaling at runtime is not within the scope of this specification.

An example of `ScaleOutSection` is as follows:

```
<VirtualSystemCollection ovf:id="web-tier">
...
 <ovf:ScaleOutSection ovf:id="web-server">
  <Info>Web tier</Info>
  <ovf:Description>Number of web server instances in web tier</ovf:Description>
  <ovf:InstanceCount ovf:default="4" ovf:minimum="2" ovf:maximum="8"/>
 </ovf:ScaleOutSection>
...
 <VirtualSystem ovf:id="web-server">
  <Info>Prototype web server</Info>
  ...
 </VirtualSystem>
</VirtualSystemCollection>
```

This section needs to be handled in two parts. The first part is to prompt the user to select the appropriate configuration. Secondly, depending on the configuration selected, replicate those number of VMs as OS::Nova::Server in the `resource_def` property of the OS::Heat::ResourceGroup resource.

The `count` property of the `OS::Heat::ResourceGroup` helps to boot the server “n” number of times.

Example for scaling on server:

```
parameters:
  param_1:
    constraints:
    - range:
        max: '8'
        min: '2'
    default: '4'
    description: number of vms to scale
    label: Number of web_sever's to scale
    type: number
resources:
  web_server:
    properties:
      count:
        get_param: param_1
      resource_def:
        properties:
          ...
        type: OS::Nova::Server
    type: OS::Heat::ResourceGroup
```

4.1.17 PlacementGroupSection and PlacementSection Interpretation

Guest software may require the deployment of virtual systems with specific proximity needs. There are two use cases:

- The ability to specify that two or more virtual systems must be deployed closely together because they rely on fast communication or have a common dependency
- The ability to specify that two or more virtual systems must be deployed on different platforms or locations because of high-availability or disaster recovery considerations

The `PlacementGroupSection` element allows an OVF package to define a placement policy for a group of `VirtualSystem`s. The `PlacementSection` element allows the annotation of the elements with membership of a particular placement policy group.

An example `PlacementSection` is as follows:

```
<Envelope>
...
<ovf:PlacementGroupSection ovf:id="AFFINITY" ovf:policy="affinity">
  <Info>Placement policy for affinity group</Info>
  <ovf:Description>Placement policy for affinity group</ovf:Description>
</ovf:PlacementGroupSection>
<ovf:PlacementGroupSection ovf:id="AVAILABILITY" ovf:policy="availability">
  <Info>Placement policy for group</Info>
   <ovf:Description>Placement policy for group</ovf:Description>
</ovf:PlacementGroupSection>
...
<StartupSection>
   <Item ovf:id="edccfc_Cirros-1" ovf:order="1" ovf:startAction="powerOn" ovf:startDelay="0" ⇒
ovf:stopAction="powerOff" ovf:stopDelay="0" ovf:shutdownorder="3"/>
   <Item ovf:id="edccfc_Cirros-2" ovf:order="2" ovf:startAction="powerOn" ovf:startDelay="60" ⇒
ovf:stopAction="powerOff" ovf:stopDelay="0" ovf:shutdownorder="2"/>
   <Item ovf:id="edccfc_Cirros-3" ovf:order="3" ovf:startAction="powerOn" ovf:startDelay="120" ⇒
ovf:stopAction="powerOff" ovf:stopDelay="0" ovf:shutdownorder="1"/>
</StartupSection>
<VirtualSystem ovf:id="Cirros-1">
...
<ovf:PlacementSection ovf:group="AFFINITY">
   <Info>Placement policy group: AFFINITY</Info>
</ovf:PlacementSection>
<ovf:PlacementSection ovf:group="AVAILABILTY">
   <Info>Placement policy group: AVAILABILITY</Info>
</ovf:PlacementSection>
...
</VirtualSystem>
<VirtualSystem ovf:id="Cirros-2">
...
<ovf:PlacementSection ovf:group="AFFINITY">
   <Info>Placement policy group: AFFINITY</Info>
</ovf:PlacementSection>
...
</VirtualSystem>
<VirtualSystem ovf:id="Cirros-3">
...
<ovf:PlacementSection ovf:group="AVAILABILITY">
   <Info>Placement policy group: AVAILABILITY</Info>
</ovf:PlacementSection>
...
</VirtualSystem>
</VirtualSystemCollection>
</Envelope>
```

`OS::Nova::Server` has a property, scheduler hints, which helps to boot a server by providing hints regarding a reference server.

An example for availability is as follows:
`Cirros-1:`

```
    properties:
    ...
      scheduler_hints:
        group:
          get_resource: AVAILABILITY
    type: OS::Nova::Server
Cirros-2:
    properties:
      ...
      scheduler_hints:
        group:
          get_resource: AFFINITY
    type: OS::Nova::Server
Cirros-3:
    properties:
      ...
      scheduler_hints:
        group:
          get_resource: AVAILABILITY
    type: OS::Nova::Server
AFFINITY:
    properties:
      name: AFFINITY
      policies:
      - AFFINITY
    type: OS::Nova::ServerGroup
AVAILABILITY:
    properties:
      name: AVAILABILITY
      policies:
      - ANTI-AFFINITY
    type: OS::Nova::ServerGroup
```

The `server group = availability/availability-host` makes sure that instances are launched on different compute nodes.

Similarly, `server gorup = affinity/affinity-host` makes sure that instances are launched on the same compute nodes.

The placement policies for affinity and availability are translated to `OS::Nova::ServerGroup`.

`OS::Nova::Server` (property – scheduler_hints)

4.1.18 EncryptionSection Interpretation

OVF specifies the use of XML encryption 1.1 standards for encrypting the files in the `ReferenceSection` or any part of the OVF descriptor file.

If such an encryption is present in the package, then this must be decrypted before proceeding with other functionalities. Currently `EncryptionSection` is not supported.

4.1.19 VirtualHardwareSection Interpretation

A `VirtualHardwareSection` element contains child elements that describe the virtual system and the virtual hardware resources (CPU, memory, network, and storage).

A `VirtualHardwareSection` element must have the following direct child elements:

- Zero or one `System` elements
- Zero or more `Item` elements
- Zero or more `EthernetPortItem` elements
- Zero or more `StorageItem` elements

The `System` element is an XML representation of the values of one or more properties of the CIM class CIM_VirtualSystemSettingData.

The virtual hardware characteristics are described as a sequence of `Item` elements.

The network hardware characteristics are described as a sequence of `EthernetPortItem` elements.

The storage hardware characteristics are described as a sequence of `StorageItem` elements.

An example of `VirtualHardwareSection` is as follows:

```
<VirtualHardwareSection>
        <Info>Virtual hardware requirements</Info>
        <System>
          <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
          <vssd:InstanceID>0</vssd:InstanceID>
          <vssd:VirtualSystemIdentifier>PL7</vssd:VirtualSystemIdentifier>
        </System>
        <Item>
          <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits>
          <rasd:Description>Number of Virtual CPUs</rasd:Description>
          <rasd:ElementName>6 virtual CPU(s)</rasd:ElementName>
          <rasd:InstanceID>1</rasd:InstanceID>
          <rasd:ResourceType>3</rasd:ResourceType>
          <rasd:VirtualQuantity>6</rasd:VirtualQuantity>
        </Item>
        <Item>
          <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
          <rasd:Description>Memory Size</rasd:Description>
          <rasd:ElementName>8192MB of memory</rasd:ElementName>
          <rasd:InstanceID>2</rasd:InstanceID>
          <rasd:ResourceType>4</rasd:ResourceType>
          <rasd:VirtualQuantity>8192</rasd:VirtualQuantity>
        </Item>
        <Item>
          <rasd:Address>1</rasd:Address>
          <rasd:Description>IDE Controller</rasd:Description>
          <rasd:ElementName>IDE 1</rasd:ElementName>
          <rasd:InstanceID>3</rasd:InstanceID>
          <rasd:ResourceType>5</rasd:ResourceType>
        </Item>
        <Item>
          <rasd:Address>0</rasd:Address>
          <rasd:Description>IDE Controller</rasd:Description>
          <rasd:ElementName>IDE 0</rasd:ElementName>
          <rasd:InstanceID>4</rasd:InstanceID>
          <rasd:ResourceType>5</rasd:ResourceType>
        </Item>
        <Item ovf:required="false">
          <rasd:AutomaticAllocation>false</rasd:AutomaticAllocation>
          <rasd:ElementName>Video card</rasd:ElementName>
          <rasd:InstanceID>5</rasd:InstanceID>
          <rasd:ResourceType>24</rasd:ResourceType>
        </Item>
        <EthernetPortItem>
       <epasd:Address>00:50:56:01:01:0d</epasd:Address>
          <epasd:AddressOnParent>7</epasd:AddressOnParent>
          <epasd:AutomaticAllocation>true</epasd:AutomaticAllocation>
          <epasd:Connection>internal-0</epasd:Connection>
          <epasd:Description>E1000 ethernet adapter on "internal-0"⇒
</epasd:Description>
          <epasd:ElementName>Network adapter 1</epasd:ElementName>
          <epasd:InstanceID>8</epasd:InstanceID>
          <epasd:ResourceSubType>E1000</epasd:ResourceSubType>
          <epasd:ResourceType>10</epasd:ResourceType>
          <ecs:staticIPAddress>10.11.96.5</ecs:staticIPAddress>

        </EthernetPortItem>
      </VirtualHardwareSection>
```

Each resource has a unique resource type.

The data regarding frequency, RAM, and CPUs are used for flavor creation (along with the details in Disk Section).

The data regarding the Network adaptor is used in port creation on the network where the VM is launched.

Flavor Characters →`Ericsson::Nova::Flavor`

MAC/IP details →`OS::Neutron::Port` or `Ericsson::Neutron::Port`

4.1.20 ReferenceSection Interpretation

`ReferenceSection` contains file references of all files that are used during deployment – images, AEFs.

An example of `ReferenceSection`, where the files are images used for deploying instances, is as follows:

```
<References>
   <File ovf:href="cscf-disk1.vmdk" ovf:id="file1" ovf:size="4023003452" />
   <File ovf:href="cscf-disk2.vmdk" ovf:id="file2" ovf:size="345232" />
</References>
```

These files are again referred in the Virtual System section by the `Item` element (of Resource type 17) in `VirtualHardware` Section. In HOT conversion, this data is used by the resource **OS::Nova::Server** under the property `Image`.

Another example of `ReferenceSection`, where the files are AEFs used for post-installation by the intended VMs, is as follows:

```
<References>
   <File ovf:id="config" ovf:href="config.xml" ovf:size="4332"/>
   <File ovf:id="resources" ovf:href="http://mywebsite/resources/resources.zip"/>
</References>
```

These files are again referred in the Virtual System section by `EnvironmentFilesSection`. In HOT conversion, this data is used by the resource `OS::Nova::Server` under the property `Personality`.

4.1.21 VirtualSystemSection Interpretation

The `VirtualSystem` element describes a single virtual system and is a container of section elements. These section elements describe virtual hardware, resources, and product information.

The `VirtualSystem` section basically encapsulates `VirtualHardware` Section and various core metadata sections as described in Table 1.

Hence each section within the `VirtualSystem` is handled separately, with the majority of interpretation leading to correspondence with the `OS::Nova::Server`.

4.1.22 ResourceGroupSection Interpretation

The extensions for this section define the `ecs:ResourceGroup` section, which is used when creating assets. Resource groups are used to group and isolate assets in availability zones.

When a VM is created, it can be associated with a particular availability zone, so that it is created in one of the servers with the chosen availability zone.

An example of `ecs:ResourceGroup` section is as follows:

```
<ecs:ResourceGroupSection>
         <Info>Resource Group Section</Info>
         <ecs:ResourceGroup ecs:id="Group1">
                   <ecs:Placement ecs:name="placementZone" ecs:value="nova">
         </ecs:ResourceGroup>
         <ecs:ResourceGroup ecs:id="Group2">
                  <ecs:Placement ecs:name="placementZone" ecs:value="nova2">
         </ecs:ResourceGroup>
   </ecs:ResourceGroupSection>
```

In `ecs:ResourceGroupSection`, `ecs:value` is mapped to the `availability_zone` property in `OS::Nova::Server` resource type.

4.2 OVF Package Validation

If a manifest file and certificate file are present in the OVF, then it is mandatory that the package is validated against those files and also the images described in the OVF file must be present in the OVF package.

This can be achieved in the pre-deployment phase before uploading the images.

Note: There is a limitation posed by OpenStack for booting an instance without an image. Thus to boot a VM from the network, specify an empty image named “ipxe.iso” in the OVA package.

The digests present must be validated against the algorithm used (SHA1 or SHA256).

5 Open Issues

The following open issues need to be studied from all aspects – configuration of CEE to implementation of Deployment Functionalities:

Annotation Section

Passing the information to the **Description** section in the **Instance Overview** tab

EULA Section

When, or if, the licensing part must be handled

Boot Device Section

OpenStack enhancements needed for Network and CD ROM booting

Shared Disk Section

CEE configuration for shared storage

Prompting Users

Work on how users must be prompted for the necessary sections - Horizon and pre-deployment module Interworking

6 OVF to HOT Map Details

A matrix of OVF to HOT mapping and OVF specification handling is shown in Table 2.

Table 2 OVF to HOT Mapping

Section Element	Parent Element	HOT Interpretation	Comments	Further Details
DiskSection	Envelope	Yes	Ericsson::Nova::Flavor (property: disk)	See Section 6.2 on page 30.
NetworkSection	Envelope	Yes	OS::Neutron::Net, OS::Neutron::ProviderNet, OS::Neutron::Subnet, Ericsson::Neutron::Port	See Section 6.3 on page 31.
ResourceGroupSection	Envelope	Yes	OS::Nova::Server (property: availability_zone)	See Section 6.4 on page 36
VirtualRouterSection	Envelope	Yes	OS::Neutron::Router	See Section 6.5 on page 37.
VirtualNetworkRouterInterfaceSection	Envelope	Yes	OS::Neutron::RouterInterface	See Section 6.6 on page 38.
ResourceAllocationSection	VirtualSystemCollection	No	Must be handled as VirtualHardware Section.	
AnnotationSection	VirtualSystem VirtualSystemCollection	No	Description exists for instances. Should find a way to add it there.	
ProductSection	VirtualSystem VirtualSystemCollection	Yes	OS::Nova::Server (property – personality:Map)	See Section 6.7 on page 39
EulaSection	VirtualSystem VirtualSystemCollection	No	Should be handled in the pre-deployment module.	
StartupSection	VirtualSystemCollection	Yes	Ericsson::Heat::DelayPolicy	See Section 6.8 on page 40.
DeploymentOptionSection	Envelope	Yes	Based on the configuration selected, relevant translation must be made.	See Section 6.9 on page 41.
OperatingSystemSection	VirtualSystem	No	Look at the possibility of adding this info as Image Metadata during pre-deployment.	See Section 6.11 on page 44.
InstallSection	VirtualSystem	No	OEFs can be handled as in case of Product Section. Instance shut down needs to be handled separately.	

Table 2 OVF to HOT Mapping

Section Element	Parent Element	HOT Interpretation	Comments	Further Details
EnvironmentFilesSection	VirtualSystem	Yes	OS::Nova::Server (property – personality:Map).	See Section 6.12 on page 44.
BootDeviceSection	VirtualSystem	No	OpenStack needs enhancements to be able to handle different booting options.	See Section 6.14 on page 46.
SharedDiskSection	Envelope	No	Depends on the OpenStack / CEE configuration.	
ScaleOutSection	VirtualSystemCollection	Yes	OS::Nova::Server (Property:count)	
PlacementGroupSection	Envelope	Yes	OS::Nova::Server (property – scheduler_hints) OS::Nova::Server (property – availability_zone) Needs more analysis on the criteria to select zones.	See Section 6.13 on page 45.
PlacementSection	VirtualSystemVirtualSystemCollection	Yes		
EncryptionSection	Envelope	No	Should be handled in the pre-deployment module based on XML encryption 1.1 standards.	
VirtualHardwareSection	VirtualSystem	Yes	Flavor characters--Ericsson::Nova::Flavor Mac details--OS::Neutron::Port/Ericsson::Neutron::Port	See Section 6.15 on page 46.
ReferenceSection	Envelope	Yes	OS::Nova::Server (property – image, personality)	See Section 6.1 on page 29.
VirtualSystemSection	Envelope VirtualSystemCollection	Yes	Collection of various sections mentioned above	See Section 6.15 on page 46.

6.1 ReferenceSection Details

The `VirtualHardwareSection` of each VM includes the `diskid` (in the `DiskSection`) that refers to the image files in this section. This file name is used in the `image` property for the Nova Server resource for that VM.

The `ecs:existing` option sets the already existing image to be used from Glance.

Example:

```
<References>
    <File ovf:href="cscf-disk1.vmdk" ovf:id="file1" ovf:size="4023003452" />
    <File ovf:href="cscf-disk2.vmdk" ovf:id="file2" ovf:size="4023003452" />
  </References>
```

HOT Mapping

```
Instance_1:
   properties:
     flavor: {get_resource: flavor_SC1}
     image: cscf-disk1.vmdk
     name: SC1
   type: OS::Nova::Server
```

6.2 DiskSection Details

The attribute `ovf:capacity` represents the disk size. The disk size is used for flavor selection of creation. The `OS::Nova::Flavor` resource is created with the required flavor details and it is referenced from the `OS::Nova::Server` resource.

The attribute `ecs:volumeType` represents the volume type of block storage volume. This is referred from the `VirtualHardware` section to differentiate a volume create based on `volumeType=SATA/SSD`. In case Disk has a `volumeType=SATA/SSD` and fileRef tag for an image then bootable volume is created. If fileRef is not present for volume creation then an empty volume is created from the `OS::Cinder::Volume` resource.

Example:

```
<DiskSection>
   <Info>Virtual disk information</Info>
   <Disk ovf:capacity="200" ovf:capacityAllocationUnits="byte * 2^30"⇒
 ovf:diskId="vmdisk1" ovf:fileRef="file1"⇒
 ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#streamOptimized" />
   <Disk ovf:capacity="50" ovf:capacityAllocationUnits="byte * 2^30"⇒
 ecs:volumeType="SATA" ovf:diskId="bsv1"  ovf:fileRef="file2"⇒
 ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#streamOptimized" />
   <Disk ovf:capacity="50" ovf:capacityAllocationUnits="byte * 2^30"⇒
 ecs:volumeType="SATA" ovf:diskId="bsv1"  ovf:fileRef="file2"⇒
 ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#streamOptimized" />
   </DiskSection>
```

HOT Mapping

```
flavor_SC1:
  properties:
    disk: 200
    ram: 8192
    vcpus: 6
  type: Ericsson::Nova::Flavor
flavor_SC2:
  properties:
     disk: 0
     ram: 8192
```

```
      vcpus: 6
  type: Ericsson::Nova::Flavor
Instance_1:
  properties:
    block_device_mapping:
      - device_name: /dev/vdb
        volume_id:
          get_resource: SC1_Volume_resource
    config_drive: 'True'
    flavor: {get_resource: flavor_SC1}
    image: cscf-disk1.vmdk
    name: SC1
    networks:
    - port: {get_resource: port_5}
  type: OS::Nova::Server
Instance_2:
    properties:
      block_device_mapping:
      - device_name: vda
        volume_id:
          get_resource: SC2_Volume_resource
      config_drive: 'True'
      flavor:
        get_resource: flavor_SC2
      name: SC2
            networks:
        - port: {get_resource: port_5}
    type: OS::Nova::Server
SC1_Volume_resource:
    properties:
      name: SC1_Volume
      size: 50
    type: OS::Cinder::Volume
SC2_Volume_resource:
    properties:
      image: 806726be-d422-4e48-8b31-6164f767261a
      name: SC2_Volume
      size: 50
    type: OS::Cinder::Volume
```

6.3 NetworkSection Details

The attribute `ovf:name` represents the network name. Each network element must be given a unique name by using the `ovf:name` attribute. The `ovf:name` is a required attribute in `NetworkSection`, while other attributes are optional. The name of the `ovf:name` attribute must be unique within an OVF envelope. An `OS::Neutron::Net` resource is created with the network name, and an `OS::Neutron::Subnet` resource is created with the details of `ecs:enableDHCP` and `ecs:IPAddressRange`.

The ecs:net_name attribute defines the network and subnet name. If this attribute is not present, ovf:name is used to define the network and subnet name.

The ecs:existing attribute defines that the network already exists in Neutron. If this attribute is used, a new network resource is not created in the HOT template and a parameter is created with default value as the existing Neutron network ID. When a Neutron network with the same name is not found, a parameter is created with empty default value.

The ecs:staticIPAddressRange or ecs:IPAddressRange attribute defines the subnet range of the network.

The ecs:dhcpEnabled or ecs:enableDHCP attribute specifies if DHCP is enabled or not.

The ecs:external attribute defines if the network being created is external or not. The default value is false.

The ecs:category attribute defines if the network is Layer2 or Layer3. The default value is Layer3. This attribute is applicable only if ecs:external="false".

Note: If ecs:external or ecs:category is not set, the translated network will have router:external=true for resource OS::Neutron::Net.

The ecs:networkType attribute identifies the type of network. Valid values include:

- local
- flat
- vlan
- gre
- vxlan

In HOT only flat and vlan are supported. The OS::Neutron::Net resource is created with the provider:network_type property.

The ecs:physicalNetwork attribute is used to define the name of a physical network used by tenants. The OS::Neutron::Net resource is created with the provider:physical_network property.

Note: ecs:useEcmSegmentationId = "true" is not supported by Atlas, as the attribute is ECM-specific. Only ecs:useEcmSegmentationId = "false" is supported in Atlas.

The ecs:segmentationId attribute is used to define the segmentation ID for a network along with the ecs:physicalNetwork and the ecs:networkType

with appropriate values, and the `ecs:useEcmSegmentationId` set to `false`. The OS::Neutron::ProviderNet resource is created with the `provider:segmentationId`.

For every unique VLAN ID in the Allowed to Transmit/Receive attribute, ECM submits a create request for each subport and also submits a create network request for the connecting Network. These represent the trunk subports to network mappings.

The networks that are created from this notation are generated by concatenating the Parent Network Name with the Application ID value from the Allowed to Transmit/Receive VLAN ID.

`NetworkPortProfile` is used to group the VLAN subports.

`AllowedToTransmitVLANs`/`AllowedToReceiveVLANs` identifies the Application VID value.

Example:

```
<DiskSection>
    <Info>Virtual disk information</Info>
    <Disk ovf:capacity="200" ovf:capacityAllocationUnits="byte * 2^30"⇒
 ovf:diskId="vmdisk1" ovf:fileRef="file1"⇒
 ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#streamOptimized" />
    <Disk ovf:capacity="50" ovf:capacityAllocationUnits="byte * 2^30"⇒
 ecs:volumeType="SATA" ovf:diskId="bsv1"  ovf:fileRef="file2"⇒
 ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#streamOptimized" />
    <Disk ovf:capacity="50" ovf:capacityAllocationUnits="byte * 2^30"⇒
 ecs:volumeType="SATA" ovf:diskId="bsv1" ovf:fileRef="file2"⇒
 ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#streamOptimized" />
  </DiskSection>
<NetworkSection>
<Network ovf:name="demo">
            <Description>The demo network</Description>
            <NetworkPortProfile>
            <Item>
                <epasd:AllowedToTransmitVLANs>101</epasd:AllowedToTransmitVLANs>
                <epasd:AllowedToReceiveVLANs>101</epasd:AllowedToReceiveVLANs>
                <epasd:AllowedToTransmitVLANs>102</epasd:AllowedToTransmitVLANs>
                <epasd:AllowedToReceiveVLANs>102</epasd:AllowedToReceiveVLANs>

           </Item>
        </NetworkPortProfile>
</Network>
<Network ovf:name="Layer2_Network"
        ecs:net_name="Layer2_Network_name"
        ecs:external="false"
        ecs:category="layer2"
        ecs:enableDHCP="true"
        ecs:networkType="vlan"
        ecs:physicalNetwork="default"
        ecs:IPAddressRange="10.33.168.64/27"
        ecs:useEcmSegmentationId="False"
        ecs:segmentationId="3950">
    <Description>The mtas-002-538-layer2-0 network</Description>
</Network>
<Network ovf:name="Layer3_Network1"
        ecs:external="false"
        ecs:category="layer3"
        ecs:enableDHCP="true"
        ecs:networkType="vlan"
        ecs:physicalNetwork="default"
        ecs:IPAddressRange="12.41.0.16/29"
        ecs:useEcmSegmentationId="False"
        ecs:segmentationId="3982">
    <Description>The mtas-002-538-external network</Description>
</Network>
 </NetworkSection>
```

HOT Mapping

```
demo:
     properties:
     name: demo
   type: OS::Neutron::Net
 demo_101:
   properties:
     name: demo_101
   type: OS::Neutron::Net
 demo_101_subnet:
   properties:
     cidr:
       get_param: param_1
     gateway_ip:
       get_param: param_2
     name: demo_101_subnet
```

```
    network_id:
      get_resource: demo_101
  type: OS::Neutron::Subnet
demo_102:
  properties:
    name: demo_102
  type: OS::Neutron::Net
demo_102_subnet:
  properties:
    cidr:
      get_param: param_3
    gateway_ip:
      get_param: param_4
    name: demo_102_subnet
    network_id:
      get_resource: demo_102
  type: OS::Neutron::Subnet
demo_subnet:
  properties:
    cidr:
      get_param: param_5
    enable_dhcp: true
    gateway_ip:
      get_param: param_6
    name: demo_subnet
    network_id:
      get_resource: demo
  type: OS::Neutron::Subnet
port_1:
    properties:
    network_id:
      get_resource: demo
    trunkport:
      type: trunk
  type: Ericsson::Neutron::Port
port_2:
  properties:
    network_id:
      get_resource: demo_101
    trunkport:
      parent_id:
        get_resource: port_1
      type: subport
      vid: 101
  type: Ericsson::Neutron::Port
port_3:
  properties:
    network_id:
      get_resource: demo_102
    trunkport:
      parent_id:
```

```
        get_resource: port_1
      type: subport
      vid: 102
  type: Ericsson::Neutron::Port
Layer2_Network:
   properties:
    name: Layer2_Network_Name
    network_type: vlan
    physical_network: default
    segmentation_id: '3950'
  type: OS::Neutron::ProviderNet
Layer2_Network_subnet:
  properties:
    cidr: 10.33.168.64/27
    enable_dhcp: true
    name: Layer2_Network_Name_subnet
    network_id:
      get_resource: Layer2_Network
  type: OS::Neutron::Subnet
Layer3_Network1:
   properties:
      name: Layer3_Network1
    value_specs:
      provider:network_type: vlan
      provider:physical_network: default
      provider:segmentation_id: '3982'
      router:external: true
  type: OS::Neutron::Net
Layer3_Network1_subnet:
  properties:
    cidr: 12.41.0.16/29
    enable_dhcp: true
    name: Layer3_Network1_subnet
    network_id:
      get_resource: Layer3
  type: OS::Neutron::Subnet
```

6.4 ResourceGroupSection Details

The extensions for this section define the `ecs:ResourceGroup` section, which is used when creating assets. Resource groups are used to group and isolate assets in availability zones. When a VM is created, it may be associated with a particular availability zone so that it is created in one of the servers with the chosen availability zone.

An example is:

```
<Envelope>
 ...
  <ecs:ResourceGroupSection>
         <Info>Resource Group Section</Info>
         <ecs:ResourceGroup ecs:id="Group1">
                   <ecs:Placement ecs:name="placementZone" ecs:value="nova">
                   </ecs:Placement>
         </ecs:ResourceGroup>
         <ecs:ResourceGroup ecs:id="Group2">
                  <ecs:Placement ecs:name="placementZone" ecs:value="nova2">
                  </ecs:Placement>
         </ecs:ResourceGroup>
   </ecs:ResourceGroupSection>
...
<VirtualSystem ovf:id="VM1" ecs:resourceGroupRef="Group1">
......
</VirtualSystem>
<VirtualSystem ovf:id="VM2" ecs:resourceGroupRef="Group2">
.....
</VirtualSystem>
...
</Envelope>
```

HOT Mapping

```
VM1:
  properties:
    ...
    availability_zone: nova
  type: OS::Nova::Server
VM2:
  properties:
    ...
    availability_zone: nova2
  type: OS::Nova::Server
```

6.5 VirtualRouterSection Details

This extension section defines ECM virtual routers (VRFs) in the OVF file. This top-level element contains a list of new Virtual Router elements.

The `ecs:id` attribute uniquely identifies a virtual router within the OVF. The `ecs:id` is used for creating the virtual router name.

Example:

```
<ecs:VirtualRouterSection>
  <Info>Virtual Router Section</Info>
  <ecs:VirtualRouter
    ecs:id="vfr1"
    ecs:name="VRF_OVF1"/>
</ecs:VirtualRouterSection>
```

HOT Mapping:

```
vrf1:
```

```
    depends_on:
    - Layer3_Network1_bgw_port
    properties:
      name: vrf1
    type: OS::Neutron::Router
```

6.6 VirtualNetworkRouterInterfaceSection Details

This extension section defines ECM Virtual Router Network Interface Cards (VRF VNICs) in the OVF file This top-level element contains a list of new virtual router elements.

Example:

```
<ecs:VirtualNetworkRouterInterfaceSection>
    <Info>VN Router Interface Section</Info>
    <ecs:VirtualNetworkRouterInterface
       ecs:id="vfr_vnic1"
       ecs:name="VRF_VNIC_OVF1"
       ecs:category="internalVrfInterface"
       ecs:virtualRouterRef="vfr1"
       ecs:networkRef="Network_OVF"/>
</ecs:VirtualNetworkRouterInterfaceSection>
<ecs:VirtualNetworkRouterInterfaceSection>
    <Info>VN Router Interface Section</Info>
    <ecs:VirtualNetworkRouterInterface
       ecs:id="bgw_port1"
       ecs:name="bgw_port1"
       ecs:category="gatewayVrfInterface"
       ecs:borderGatewayHostId="BGW-1"
       ecs:networkRef="Network_OVF"/>
</ecs:VirtualNetworkRouterInterfaceSection>
```

HOT Mapping:

```
VRF_VNIC1:
  properties:
    router_id:
      get_resource: VRF1
    subnet_id:
      get_resource: Layer3_Network1_subnet
  type: OS::Neutron::RouterInterface
bgw_port1:
  properties:
    binding:
      host_id: BGW-1
    device_owner: baremetal:BGW-1
    network_id:
```

```
            get_resource: Network_OVF
        type: Ericsson::Neutron::Port
```

6.7 ProductSection Details

The product section is converted to an environment file (OEF) and is placed in the VM based on the path mentioned.

The OEF file is used in the personality property of the OS::Nova::Server. There is no support for the usage of environment variables from the environment file after VM boot up.

Each property key-value pair in the product section is also copied into the metadata section in the properties of OS::Nova::Server.

The properties in `ProductSection` with the attribute `ovf:userConfigurable="true"` are translated as parameters which could be configured by the user at the time of stack launch. The properties with `ovf:userConfigurable="false"` are directly placed into the personality section and metadata section of the OS::Nova::Server.

Example:

```
        <ProductSection>
            <Info>Information about the installed software</Info>
            <Product>Ericsson CSCF</Product>
            <Vendor>Ericsson AB</Vendor>
            <Version>14A</Version>
            <FullVersion>14A</FullVersion>
            <Property ovf:key="application.type" ovf:type="string" ovf:userConfigurable="false" ⇒
ovf:value="CSCF">
                <Label>APPLICATION_TYPE</Label>
                <Description>APPLICATION_TYPE</Description>
            </Property>
            <Property ovf:key="domain.name" ovf:type="string" ovf:userConfigurable="true" ⇒
ovf:value="ims.ericloud">
                <Label>DOMAIN_NAME</Label>
                <Description>DOMAIN_NAME</Description>
            </Property>
            <Property ovf:key="g3.subnetwork" ovf:type="string" ovf:userConfigurable="true">
                <Label>G3_SUBNETWORK</Label>
                <Description>G3_SUBNETWORK</Description>
            </Property>
        </ProductSection>
```

HOT Mapping

```
Instance_1:
    properties:
        flavor: {get_resource: flavor_SC1}
        image: cscf-disk1.vmdk
        Name: SC1
        metadata:
            application.type: CSCF
            domain.name: get_param: param_4
            g3.subnetwork: get_param: param_3
        personality:
            /ovf-env.xml:
                str_replace:
                    params:
                        $domain.name: get_param: param_4
                        $g3.subnetwork: get_param: param_3
                    template: "<Environment xmlns:ovfenv=\"http://schemas.dmtf.org/ovf/environment/1\"\
                        \ xmlns=\"http://schemas.dmtf.org/ovf/envelope/1\">\n  <PlatformSection>\n\
                        \    <Vendor>Ericsson AB</Vendor>\n    <Product>Ericsson CSCF</Product>\n\
                        \    <FullVersion>14A</FullVersion>\n    <Version>14A</Version>\n  \
                        \  <Info>Information about the installed software</Info>\n  </PlatformSection>\n\
                        \  <PropertySection>\n    <Property ⇒ovfenv:key=\"application.type\"\
                        \ ovfenv:value=\"CSCF\"/>\n    <Property ovfenv:key=\"g3.subnetwork\"\
                        \ ovfenv:value=\"$g3.subnetwork\"/>\n    <Property ovfenv:key=\"domain.name\"\
                        \ ovfenv:value=\"$domain.name\"/>\n  </PropertySection>\n</Environment>\n"
    type: OS::Nova::Server
```

6.8 StartupSection Details

The HOT template provides the following resource which induces a delay between VM launches (one VM waits for previous one to be booted): Ericsson::Heat::DelayPolicy.

Example:

```
<StartupSection>
           <Item ovf:id="PL-3" ovf:order="2" ovf:startAction="powerOn"
 ovf:startDelay="120" ovf:stopAction="powerOff" ovf:stopDelay="120"/>
           <Item ovf:id="PL-4" ovf:order="3" ovf:startAction="powerOn"
 ovf:startDelay="120" ovf:stopAction="powerOff" ovf:stopDelay="120"/>
           <Item ovf:id="SC-2" ovf:order="4" ovf:startAction="powerOn"
 ovf:startDelay="120" ovf:stopAction="powerOff" ovf:stopDelay="120"/>
           <Item ovf:id="SC-1" ovf:order="1" ovf:startAction="powerOn"
 ovf:startDelay="150" ovf:stopAction="powerOff" ovf:stopDelay="120"/>
 </StartupSection>
```

HOT Mapping

```
 delay_1:
    depends_on:
    - SC-1
    properties:
      delay: 150
    type: Ericsson::Heat::DelayPolicy
  delay_2:
    depends_on:
    - PL-3
    properties:
      delay: 120
    type: Ericsson::Heat::DelayPolicy
  delay_3:
    depends_on:
```

```
- PL-4
properties:
  delay: 120
type: Ericsson::Heat::DelayPolicy
```

6.9 DeploymentOptionSection Details

The `DeploymentOptionSection` element specifies a discrete set of intended resource configurations. When deploying the stack, use the "default" configuration or select from the allowed configurations of the `DeploymentOptionSection` `ovf:id` attributes.

Currently, `DeployementOptionSection` is supported to configure VCPUs and RAM for a particular Virtual System.

Example:

```
...
   <DeploymentOptionSection>
       <Configuration ovf:id="deploy2">
           <Label ovf:msgid="deploy2.label">deploy2</Label>
           <Description ovf:msgid="deploy2.description">deploy2 Configuration</Description>
       </Configuration>
       <Configuration ovf:id="deploy1">
           <Label ovf:msgid="deploy1.label">deploy1</Label>
           <Description ovf:msgid="deploy1.description">deploy1 Configuration</Description>
       </Configuration>
   </DeploymentOptionSection>
 ....
 <VirtualSystem ovf:id="vm1">
    ....
               <Item>
                   <rasd:AllocationUnits>hertz*10^6</rasd:AllocationUnits>
                   <rasd:Description>Number of Virtual CPUs</rasd:Description>
                   <rasd:ElementName>4 virtual CPU(s)</rasd:ElementName>
                   <rasd:InstanceID>1</rasd:InstanceID>
                   <rasd:Reservation>0</rasd:Reservation>
                   <rasd:ResourceType>3</rasd:ResourceType>
                   <rasd:VirtualQuantity>4</rasd:VirtualQuantity>
               </Item>
               <Item>
                   <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
                   <rasd:Description>Memory Size</rasd:Description>
                   <rasd:ElementName>1024MB of memory</rasd:ElementName>
                   <rasd:InstanceID>2</rasd:InstanceID>
                   <rasd:Reservation>0</rasd:Reservation>
                   <rasd:ResourceType>4</rasd:ResourceType>
                   <rasd:VirtualQuantity>1024</rasd:VirtualQuantity>
               </Item>

               <Item ovf:configuration="deploy1">
                   <rasd:AllocationUnits>hertz*10^6</rasd:AllocationUnits>
                   <rasd:Description>Number of Virtual CPUs</rasd:Description>
                   <rasd:ElementName>2 virtual CPU(s)</rasd:ElementName>
                   <rasd:InstanceID>1</rasd:InstanceID>
                   <rasd:Reservation>1</rasd:Reservation>
                   <rasd:ResourceType>3</rasd:ResourceType>
                   <rasd:VirtualQuantity>2</rasd:VirtualQuantity>
               </Item>
               <Item ovf:configuration="deploy1">
                   <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
                   <rasd:Description>Memory Size</rasd:Description>
                   <rasd:ElementName>256MB of memory</rasd:ElementName>
                   <rasd:InstanceID>2</rasd:InstanceID>
                   <rasd:Reservation>64</rasd:Reservation>
                   <rasd:ResourceType>4</rasd:ResourceType>
                   <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
               </Item>
               <Item ovf:configuration="deploy2">
                   <rasd:AllocationUnits>hertz*10^6</rasd:AllocationUnits>
                   <rasd:Description>Number of Virtual CPUs</rasd:Description>
                   <rasd:ElementName>1 virtual CPU(s)</rasd:ElementName>
                   <rasd:InstanceID>1</rasd:InstanceID>
                   <rasd:Reservation>1</rasd:Reservation>
                   <rasd:ResourceType>3</rasd:ResourceType>
                   <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
               </Item>
               <Item ovf:configuration="deploy2">
                   <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
                   <rasd:Description>Memory Size</rasd:Description>
                   <rasd:ElementName>512MB of memory</rasd:ElementName>
                   <rasd:InstanceID>2</rasd:InstanceID>
                   <rasd:Reservation>64</rasd:Reservation>
                   <rasd:ResourceType>4</rasd:ResourceType>
                   <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
               </Item>
           </VirtualHardwareSection>
       </VirtualSystem>
...
```

HOT Mapping

```
parameters:
  config_vm1:
    constraints:
    - allowed_values:
      - default
      - deploy1
      - deploy2
    default: default
    description: Choose deployment configuration
    label: Deployment configurations for vm1
    type: string
resources:
...
  flavor_vm1:
    properties:
      disk: 2
      ram:
        Fn::Select:
        - get_param: config_vm1
        - default: 1024
          deploy1: 256
          deploy1: 512
      vcpus:
        Fn::Select:
        - get_param: config_vm1
        - default: 4
          deploy1: 2
          deploy2: 1
    type: Ericsson::Nova::Flavor
  vm1:
    properties:
    ...
      flavor:
        get_resource: flavor_vm1
    ...
    type: OS::Nova::Server
...
```

6.10 VirtualSystemSection Details

These details are used in `OS::Nova::Server`.

Example:

```
<VirtualSystem ovf:id="SC1">
      <Info>A virtual machine</Info>
      <Name>SC1</Name>
```

HOT Mapping

```
Instance_1:
  properties:
    name: SC1
  type: OS::Nova::Server
```

6.11 OperatingSystemSection Details

Not supported.

```
<OperatingSystemSection ovf:id="102">
        <Info>The kind of installed guest operating system</Info>
        <Description>Payload Processor</Description>
</OperatingSystemSection>
```

6.12 EnvironmentFilesSection Details

The files from `EnvironmentFilesSection` are placed in the VM based on the path specified in `ovf:path`. The OEF file is used in the `personality` property of the `OS::Nova::Server`.

The file with `ovf:path="cloud-init-data"` or `user-data.txt` is treated as the designated User-Data (AEF) file, which can be provided in HOT format by using the `user_data` property of the `OS::Nova::Server`.

Example:

```
<References>
        <File ovf:href="cirros-0.3.2-x86_64-disk1.img" ovf:id="file1"/>
        <File ovf:href="cirros-0.3.2-x86_64-disk2.img" ovf:id="file2"/>
        <File ovf:id="user-data" ovf:href="user-data.txt"/>
        <File ovf:id="config" ovf:href="config.xml"/>
    </References>
    ....
    <ovf:EnvironmentFilesSection ovf:required="false" ovf:transport="iso">
         <Info>Config files to be included in OVF environment</Info>
         <ovf:File ovf:fileRef="user-data" ovf:path="cloud-init-data"/>
         <ovf:File ovf:fileRef="config" ovf:path="setup/configuration.xml"/>
    </ovf:EnvironmentFilesSection>
     ......
```

HOT Mapping

```
Instance_1:
   properties:
        ...
        name: SC1
        config_drive: true
        personality:
           /ovf-env.xml : <file
                content>
          setup/configuration.xml:
                get_file: config.xml
        user_data:
           get_file: user-data.txt
```

```
        user_data_format: "RAW"
        ...
    type: OS::Nova::Server
```

6.13 PlacementGroupSection and PlacementSection Details

The `Scheduler_hints` property of the `OS::Nova:Server` helps to boot a server depending on its placement policy, that is, availability/affinity. The `Scheduler_hints` property has a prerequisite that scheduler service needs to be configured with the server group.

The list of servergroups is appended to `server_group_list`.

Example:

```
<ovf:PlacementGroupSection ovf:id="availability" ovf:policy="availability">
       <Info>Placement policy for group of VMs</Info>
       <ovf:Description>Placement policy for web tier</ovf:Description>
</ovf:PlacementGroupSection>
<ovf:PlacementGroupSection ovf:id="affinity" ovf:policy="affinity">
       <Info>Placement policy for group of VMs</Info>
       <ovf:Description>Placement policy for web tier</ovf:Description>
</ovf:PlacementGroupSection>
<VirtualSystem ovf:id="Instance1">
...
<ovf:PlacementSection ovf:group="affinity">
  <Info>Placement policy group: affinity</Info>
</ovf:PlacementSection>
<ovf:PlacementSection ovf:group="availability">
  <Info>Placement policy group: availabilty</Info>
</ovf:PlacementSection>
...
</VirtualSystem>
<VirtualSystem ovf:id="Instance2">
...
<ovf:PlacementSection ovf:group="affinity">
  <Info>Placement policy group: affinity</Info>
</ovf:PlacementSection>
...
</VirtualSystem>
<VirtualSystem ovf:id="Instance3">
...
<ovf:PlacementSection ovf:group="availability">
  <Info>Placement policy group: availability</Info>
</ovf:PlacementSection>
...
</VirtualSystem>
```

HOT Mapping

```
Instance1:
    properties:
    ...
      scheduler_hints:
        group:
          get_resource: availability
    type: OS::Nova::Server
Instance2:
    properties:
```

```
      ...
      scheduler_hints:
        group:
          get_resource: affinity
    type: OS::Nova::Server
Instance3:
    properties:
      ...
      scheduler_hints:
        group:
          get_resource: availability
    type: OS::Nova::Server
affinity:
    properties:
      name: affinity
      policies:
      - affinity
    type: OS::Nova::ServerGroup
availability:
    properties:
      name: availability
      policies:
      - anti-affinity
    type: OS::Nova::ServerGroup
```

6.14 BootDeviceSection Details

Not supported.

Example:

```
  <ovf:BootDeviceSection>
 <Info>Boot device order specification</Info>
 <bootc:CIM_BootConfigSetting>
  <bootc:Caption>SC with disk</bootc:Caption>
  <bootc:Description>Boot Sequence for node with disk</bootc:Description>
  <boots:CIM_BootSourceSetting>
   <boots:Caption>Boot from disk</boots:Caption>
   <boots:InstanceID>8</boots:InstanceID>
  </boots:CIM_BootSourceSetting>
  <boots:CIM_BootSourceSetting>
   <boots:Caption>Boot from internal-0</boots:Caption>
   <boots:InstanceID>10</boots:InstanceID>
  </boots:CIM_BootSourceSetting>
  <boots:CIM_BootSourceSetting>
   <boots:Caption>Boot from internal-1</boots:Caption>
   <boots:InstanceID>11</boots:InstanceID>
  </boots:CIM_BootSourceSetting>
 </bootc:CIM_BootConfigSetting>
  </ovf:BootDeviceSection>
```

6.15 VirtualHardwareSection Details

6.15.1 rasd:ResourceType 3 Item

The `rasd:ResourceType 3` item represents the virtual CPU (vCPU) and rasd:VirtualQuantity represents the quantity of the resource, that is, the number of vCPUs required for this VM.

The number of vCPUs is used for flavor selection or creation.

Example:

```
<Item>
    <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits>
    <rasd:Description>Number of Virtual CPUs</rasd:Description>
    <rasd:ElementName>6 virtual CPU(s)</rasd:ElementName>
    <rasd:InstanceID>1</rasd:InstanceID>
    <rasd:ResourceType>3</rasd:ResourceType>
    <rasd:VirtualQuantity>6</rasd:VirtualQuantity>
</Item>
```

HOT Mapping

```
flavor_SC1:
  properties: {disk: 200,ram: 8192, vcpus: 6}
  type: OS::Nova::Flavor
```

6.15.2 rasd:ResourceType 4 Item

The `rasd:ResourceType 4`iItem represents virtual memory and `rasd:VirtualQuantity` represents the quantity of the resource, that is, the size of RAM required for this VM.

The RAM size is used for flavor selection or creation.

Example:

```
<Item>
   <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
   <rasd:Description>Memory Size</rasd:Description>
   <rasd:ElementName>8192MB of memory</rasd:ElementName>
   <rasd:InstanceID>2</rasd:InstanceID>
   <rasd:ResourceType>4</rasd:ResourceType>
   <rasd:VirtualQuantity>8192</rasd:VirtualQuantity>
</Item>
```

HOT Mapping

```
flavor_SC1:
  properties: {disk: 200,ram: 8192, vcpus: 6}
  type: OS::Nova::Flavor
```

6.15.3 rasd:ResourceType 17 Item

The `rasd:ResourceType 17` item represents hard disk or image and `rasd:HostResource` represents the reference to the hard disk or image.

The hard disk reference is used to identify the image in the file section via disk section.

The image is used in `OS::Nova::Server`.

Note: If there is no hard disk specified then a bare image of 1GB is used.

The bare image must be available in OpenStack.

Example:

```
<Item>
  <rasd:AddressOnParent>0</rasd:AddressOnParent>
  <rasd:ElementName>Hard disk 1</rasd:ElementName>
  <rasd:HostResource>ovf:/disk/vmdisk1</rasd:HostResource>
  <rasd:InstanceID>8</rasd:InstanceID>
  <rasd:Parent>3</rasd:Parent>
  <rasd:ResourceType>17</rasd:ResourceType>
</Item>
```

HOT Mapping

```
Instance_1
  properties:
    flavor: {get_resource: flavor_SC1}
    image: cscf-disk1.vmdk
    name: SC1
    networks:
    - port: {get_resource: port_5}
    - port: {get_resource: port_6}
    - port: {get_resource: port_7}
    - port: {get_resource: port_8}
    - port: {get_resource: port_9}
  type: OS::Nova::Server
```

6.15.4 rasd:ResourceType 10 Item

The `rasd:ResourceType 10` item represents the NIC device.

It also specifies the network information of the NIC with a `rasd:Connection` tag.

The NIC is included in the networks of `OS::Nova::Server`.

The tag `rasd:Address` specifies the MAC address of the NIC device.

This is an optional tag. If the MAC address is specified then `OS::Neutron::Port` is created and it is referenced from the `OS::Nova::Server`.

Example:

```
<Item>
   <rasd:Address>00:50:56:01:04:09</rasd:Address>
   <rasd:AddressOnParent>9</rasd:AddressOnParent>
   <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
   <rasd:Connection>oam-vip-0</rasd:Connection>
   <rasd:Description>E1000 ethernet adapter on "oam-vip-0"</rasd:Description>
   <rasd:ElementName>Network adapter 4</rasd:ElementName>
   <rasd:InstanceID>13</rasd:InstanceID>
   <rasd:ResourceSubType>E1000</rasd:ResourceSubType>
   <rasd:ResourceType>10</rasd:ResourceType>
</Item>
```

HOT Mapping

```
oam-vip-0:
  properties: {name: oam-vip-0}
  type: OS::Neutron::Net


oam-vip-0_subnet:
  properties:
    cidr: 192.168.1.0/28
    gateway_ip: 192.168.1.1
    name: oam-vip-0_subnet
    network_id: {get_resource: oam-vip-0}
  type: OS::Neutron::Subnet

port_3:
  properties:
    mac_address: 00:50:56:01:04:09
    name: port_3
    network_id: {get_resource: oam-vip-0}
  type: OS::Neutron::Port

instance_1:
  properties:
    flavor: '5'
    image: bare
    name: PL7
    networks:
    - port: {get_resource: port_1}
    - port: {get_resource: port_2}
    - port: {get_resource: port_3}
    - port: {get_resource: port_4}
  type: OS::Nova::Server
```

6.15.5 ecs:AllowedAddressPairs

The `AllowedAddressPairs` attribute in `EthernetPortItem` represents the allowed address pairs for VM VNICs that identify the MAC address, IPv4 or IPv6 addresses, through which traffic is routed.

Example:

```
  <EthernetPortItem>
    <epasd:Connection>MME-Internal-2</epasd:Connection>
    <epasd:Description>external net</epasd:Description>
    <epasd:ElementName>Network adapter 1</epasd:ElementName>

<ecs:AllowedAddressPairs>⇒
AA:bb:22:11:CC:12|192.168.5.2;BB:bb:22:11:CC:13|193.168.5.3;⇒
</ecs:AllowedAddressPairs>
    <epasd:InstanceID>5</epasd:InstanceID>
    <epasd:ResourceType>10</epasd:ResourceType>
  </EthernetPortItem><>
```

HOT Mapping

```
instance1_port1:
      type: OS::Neutron::Port
      properties:
         network_id: <network id>
         allowed_address_pairs:
           - ip_address: 192.168.5.2
             mac_address: AA:bb:22:11:CC:12
           - ip_address: 192.168.5.3
             mac_address: AA:bb:22:11:CC:13
```

6.15.6 epasd:ElementName

`Element Name` in `EthernetPortItem` represents the name of the port to be created.

Example:

```
  <EthernetPortItem>
    <epasd:Connection>Internal_Network</epasd:Connection>
    <epasd:Description>The Internal_Network Network</epasd:Description>
    <epasd:ElementName>Network adapter 1</epasd:ElementName>
    <epasd:InstanceID>5</epasd:InstanceID>
    <epasd:ResourceType>10</epasd:ResourceType>
  </EthernetPortItem>
```

HOT Mapping

```
port_1:
  properties:
    name: Network adapter 1
    network_id:
    get_resource: Internal_Network
    type: OS::Neutron::Port
```

6.15.7 epasd:resourceSubType

In `EthernetPortItem`, `ResourceSubType` can be used to define SR-IOV attribute associated with the port. The `ResourceSubType` attribute is

translated to `vnic_type` property of `OS::Neutron::Port` in HOT for the following values:

- SR-IOV DIRECT
- SR-IOV MACVTAP

Example:

```
<EthernetPortItem>
  <epasd:Connection>MME-Internal-2</epasd:Connection>
  <epasd:Description>external net</epasd:Description>
  <epasd:ElementName>Network adapter 1</epasd:ElementName>
  <epasd:ResourceSubType>SR-IOV DIRECT</epasd:ResourceSubType>
  <epasd:InstanceID>5</epasd:InstanceID>
  <epasd:ResourceType>10</epasd:ResourceType>
</EthernetPortItem><>
```

HOT Mapping

```
instance1_port1:
      type: OS::Neutron::Port
      properties:
         network_id: <network id>
         vnic_type: direct
```

6.15.8 Other Items

There are other Items in the `VirtualHardware` section that deal with the Controllers and other hardware devices.

CD-ROM is currently not supported in OpenStack. OpenStack needs to be modified.

Image properties can be updated in order for the hypervisor to create the required controllers.

Not supported.

Example:

```
<Item>
  <rasd:Address>0</rasd:Address>
  <rasd:Description>SCSI Controller</rasd:Description>
  <rasd:ElementName>SCSI controller 0</rasd:ElementName>
  <rasd:InstanceID>3</rasd:InstanceID>
  <rasd:ResourceSubType>lsilogic</rasd:ResourceSubType>
  <rasd:ResourceType>6</rasd:ResourceType>
</Item>
<Item>
  <rasd:Address>1</rasd:Address>
  <rasd:Description>IDE Controller</rasd:Description>
  <rasd:ElementName>IDE 1</rasd:ElementName>
  <rasd:InstanceID>4</rasd:InstanceID>
  <rasd:ResourceType>5</rasd:ResourceType>
</Item>
<Item>
  <rasd:Address>0</rasd:Address>
  <rasd:Description>IDE Controller</rasd:Description>
  <rasd:ElementName>IDE 0</rasd:ElementName>
  <rasd:InstanceID>5</rasd:InstanceID>
  <rasd:ResourceType>5</rasd:ResourceType>
</Item>
<Item ovf:required="false">
  <rasd:AutomaticAllocation>false</rasd:AutomaticAllocation>
  <rasd:ElementName>Video card</rasd:ElementName>
  <rasd:InstanceID>6</rasd:InstanceID>
  <rasd:ResourceType>24</rasd:ResourceType>
</Item>
<Item>
  <rasd:AddressOnParent>0</rasd:AddressOnParent>
  <rasd:AutomaticAllocation>false</rasd:AutomaticAllocation>
  <rasd:ElementName>CD/DVD drive 1</rasd:ElementName>
  <rasd:InstanceID>9</rasd:InstanceID>
  <rasd:Parent>5</rasd:Parent>
  <rasd:ResourceType>15</rasd:ResourceType>
</Item>
```

Reference List

[1] *Open Virtualization Format Specification – version 2.1, http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_2.1.0.pdf*

[2] *XML encryption 1.1, http://www.w3.org/TR/2013/REC-xmlenc-core1-20130411/*