

OpenStack Orchestration API in CEE

Cloud Execution Environment

INTERWORK DESCRIPTION

Copyright

© Ericsson AB 2016–2018. 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	1
1.1	API Version	1
1.2	Document References	1
2	Supported Operations	2
2.1	Basic OpenStack Operations	2
2.2	OpenStack Extensions	2
3	Heat Plugins	3
3.1	Ericsson::Heat::DelayPolicy	3
3.2	Ericsson::Neutron::Port	4
3.3	Ericsson::Neutron::RouterInterface	6
4	Limitations and Recommendations	8
4.1	Limitations	8
4.2	Recommendations	8





1 Introduction

This document serves as an introduction to the Application Programming Interface (API) of the OpenStack Dashboard based cloud management component Atlas, in the Cloud Execution Environment (CEE).

The API is used for orchestration purposes in CEE.

The document describes the modifications made to the OpenStack Orchestration Newton version.

1.1 API Version

The Atlas orchestration API is based on OpenStack Orchestration API v1.

1.2 Document References

This section lists the official OpenStack API references.

1.2.1 API Design Base Reference

For a detailed description of the API operations and extensions of orchestration, refer to the section “Orchestration API v1” in the [OpenStack API Complete Reference](#).

This is a stored copy of the OpenStack API reference document that was the base for the development of this version of CEE.

All references to this OpenStack API made in this document are based on this specific document version. The document may include API Extensions that are not supported in this version of CEE. For details, see Section 2.2 on page 2.

Note: It is possible that the date on the front page differs from the one above, since the former shows the date on which the document was generated.

1.2.2 API Template Reference

For a detailed description of the Heat template, refer to [OpenStack Heat](#).

This is a stored copy of the OpenStack Heat template reference document that was the base for the development of this version of CEE.

Note: It is possible that the date on the front page differs from the one above, since the former shows the date on which the document was generated.



2 Supported Operations

This section contains information about the API operations and API extensions in CEE.

2.1 Basic OpenStack Operations

Atlas orchestration supports all basic Orchestration API operations.

For a detailed description of the API operations and extensions of orchestration, refer to the section “Orchestration API v1” in the [OpenStack API Complete Reference](#).

2.1.1 Limitations

Not applicable.

2.1.2 OpenStack Template Reference

CEE Orchestration supports all Template Reference.

For a detailed description of the Heat template, refer to [OpenStack Heat](#).

2.2 OpenStack Extensions

This section lists the OpenStack API extensions that are used in CEE.

CEE orchestration supports the Identity API v2.0 extensions documented in the section “Identity API v2.0 extensions” in the [OpenStack API Complete Reference](#).



3 Heat Plugins

The required properties and resource types related to Heat are implemented as a plugins. The properties required for Ericsson-specific tags are added in the plugins.

The plugins are implemented as Python modules and are described in Section 3.1 on page 3, and Section 3.2 on page 4.

Installation and Configuration

The file `/etc/heat/heat.conf` contains a list of directories where the plugins are stored. To define additional directories, use the `plugin_dirs` option in file `/etc/heat/heat.conf`.

By default the `/usr/lib/python2.7/dist-packages/heat/plugins` directory is included and searched for plugins.

To install a plugin, copy the Python modules to one of the configured plugin directories.

Note: The Heat engine must be restarted after this in order to load the new plugins.

3.1 Ericsson::Heat::DelayPolicy

Note: This plugin is deprecated in this release of CEE, use `OS::Heat::TestResource` instead.

The plugin enables the use of a `Delay Policy`, as a resource in a Heat template. The Python module for the plugin is `delay.py`.

The resource enables delay in seconds between allocation of two consequent resources. A typical usage of this resource is waiting for one VM to finish configuring itself before invoking deployment of another one.

The new resource has the type `Ericsson::Heat::DelayPolicy` with the properties shown in Table 1.

Table 1 Delay Policy Properties

Name	Type	Description
delay	Number	Delay between two resources

An `Ericsson::Heat::DelayPolicy` resource uses the properties in Table 1 to dynamically create a delay between two resources, as shown in Example 1.



```
```yaml
heat_template_version: 2013-05-23
description: Heat DelayPolicy creation example
resources:
 test_delay:
 type: Ericsson::Heat::DelayPolicy
 properties: {delay: 2}
...
```
```

Example 1 Delay Policy Template

3.2 Ericsson::Neutron::Port

Note: This plugin is deprecated in this release of CEE.

The plugin enables the use of port, as a resource in a Heat template. The Python module for the plugin is `eport.py`.

The new resource has the type `Ericsson::Neutron::Port` with the properties shown in Table 2.

Table 2 Ericsson::Neutron::Port Properties

| Name | Type | Description |
|-----------------------|--------|--|
| network_id | String | Unique identifier for the network owning the port |
| network | String | The network to which this port belongs |
| name | String | A symbolic name for this port |
| value_specs | Map | Extra parameters to include in the port object |
| admin_state_up | String | The administrative state of this port |
| fixed_ips | List | Desired IPs for this port |
| mac_address | String | MAC address for this port |
| dns_name | String | DNS name associated with this port |
| device_id | String | Device ID of this port |
| security_groups | List | Security group IDs to associate with this port |
| allowed_address_pairs | List | Additional MAC/IP address pairs allowed to pass through the port |
| device_owner | String | Name of the network owning the port |



| Name | Type | Description |
|-------------------------|---------|--|
| replacement_policy | String | Policy on how to respond to a stack-update for this resource.

REPLACE_ALWAYS will replace the port regardless of any property changes. AUTO will update the existing port for any changed update-allowed property. |
| trunkport | MAP | Trunk port properties |
| 1) trunk_port_type | String | Defines if a port is a trunk port or sub port |
| 2) trunk_port_parent_id | String | For sub ports the ID of the trunk port that sub port is connected to |
| 3) vid | String | For sub ports VID that will be used to access this sub port from trunk port |
| binding | MAP | Binding port properties |
| 1) host_id | String | Defines host ID of a port |
| binding:vnic_type | String | The vNIC type needs to be bound on the Neutron port. To support SR-IOV PCI passthrough networking request the neutron port to be realized as normal (vNIC), direct (PCI passthrough), or macvtap (a virtual interface with a tap-like software interface).

This procedure only works for Neutron deployments that support the bindings extension. |
| qos_policy | String | The name or ID of QoS policy to be attached to this port |
| port_security_enabled | Boolean | Flag to enable or disable port security on this port. When disabled (set to False), there is no package filtering (for example security_groups or address-pairs). |

An `Ericsson::Neutron::Port` resource uses the properties in Table 2 to dynamically create a port, as shown in Example 2.



```
```yaml
heat_template_version: 2013-05-23
description: Heat Port creation example
resources:
 test_eport:
 type: Ericsson::Neutron::Port
 properties:
 network_id: net1234
 trunkport: {
 type: subport,
 parent_id: s0e951a6-h6b4-3f4c-8c88-b95e2a120b7m',
 vid: 101
 }
...
```
```

Example 2 HOT Syntax for Ericsson::Neutron::Port Resource

3.3 Ericsson::Neutron::RouterInterface

This is a resource representation of an internal network interface to the specified router.

The new resource has the type `Ericsson::Neutron::RouterInterface` with the properties shown in Table 3.

Table 3 RouterInterface Properties

| Name | Type | Description |
|------------------|--------|---|
| router | String | Name of the router. |
| router_id | String | ID of the router. |
| port | String | Port name: either the subnet or the port should be specified. |
| port_id | String | ID of the port. |
| subnet | String | Subnet name: either the subnet or the port should be specified. |
| subnet_id | String | ID of the subnet. |
| active_device_id | String | The active device ID.
Updates cause replacement.
Optional property. |

An `Ericsson::Neutron::RouterInterface` resource uses the properties in Table 3, as shown in Example 3, using HOT syntax.



```
heat_template_version: 2013-05-23
...
resources:
  ...
  the_resource:
    type: OS::Neutron::RouterInterface
    properties:
      port: String
      router: String
      subnet: String
      active_device_id: String
```

Example 3 HOT Syntax for Router Interface Resource



4 Limitations and Recommendations

This section describes CEE specific limitations and recommendations.

4.1 Limitations

Not applicable.

4.2 Recommendations

Not applicable.