

Atlas CLI End User Guide

Cloud Execution Environment

USER GUIDE

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
2	cApp Create	3
2.1	Create a cApp	3
2.2	Create a cApp with All Option Fields	5
3	cApp Delete	8
4	cApp Update	9
5	cApp Show	11
6	cApp Show Template	13
7	List Available cApps	16
8	List Files for a cApp	17
9	Update File Content	18
10	Show File Content	21
11	cApp Export	22
12	Application Scaling	23
12.1	Stack Scale Out	23
12.2	Stack Scale In	25
12.3	Stack Scaling List	26
13	cApp Personalize	27
13.1	Adding Properties From CLI	27
13.2	cApp Personalize for Editing the Main Template	29





1 Introduction

This user guide provides the syntax descriptions and examples of the commands used for managing Atlas by using the Command-Line Interface (CLI).

The CLI can be reached by using either of the following clients:

- OVFT client
- OpenStack client

Note: The OpenStack client is planned to be the officially supported client on a long term basis.

As the current version of Atlas is based on OpenStack Newton, the following limitations apply to executing commands from Atlas CLI, compared to using OpenStack Client from any vCIC:

- Keystone commands are not available. It is recommended to use OpenStack client commands instead.
- Some OpenStack commands return a notice in the printout on the deprecation of the command.

Open Virtualization Format Translator (OVFT) Cloud Application (cApp) is an Ericsson service that provides catalog application handling. The templates stored in the Atlas database can be deployed by using the Heat Orchestration service. The following input formats are supported:

Open Virtualization Format (OVF):

The OVF package is processed to generate a corresponding HOT template and store the generated HOT in the Atlas database.

Heat Orchestration Template (HOT):

The input HOT template and its inner templates are stored in the Atlas database. Nested templates are also supported.

Topology and Orchestration Specification for Cloud Applications (TOSCA):

The input TOSCA template is stored in the Atlas database. TOSCA CSAR (Cloud Service Archive) and Tacker properties are also supported.

Note: Atlas includes Mistral, so application life cycle management is enriched with workflows and OpenStack specific actions. For further information, refer to the [OpenStack End User Guide](#).

The target group of this user guide consists of Atlas end users managing and uploading catalog cApps by using the CLI.



Note: In this document, memory and storage quantities are represented according to the JESD100B.01 standard:

- KB refers to 2^{10} bytes
- MB refers to 2^{20} bytes
- GB refers to 2^{30} bytes



2 cApp Create

This section describes how to upload cApps.

2.1 Create a cApp

Syntax for OVFT client:

```
ovft capp-create --name <capp_name> =>
--file <path_to_the_capp> --type <ovf/hot/tosca>
```

Syntax for OpenStack client:

```
openstack capp create --name <capp_name> =>
--file <path_to_the_capp> --type <ovf/hot/tosca>
```

Examples:

```
root@atlas:~# ovft capp-create --type ovf --name test-capp --file ~/test.ova
```

Property	Value
created_at	2015-12-02T06:53:49.000000
description	
fault	
id	4ebaef72-b8cc-4616-8b49-e5563fdaf0b3
image_ids	
is_protected	False
is_public	False
name	test-capp
status	Creating
type	OVF
updated_at	2015-12-02T06:53:50.000000
user_id	9a89e98b6b764a37a83ba44a27161352

Example 1 Create a cApp with Type “ovf” by Using an OVFT Client

```
root@atlas:~# openstack capp create --type ovf --name test_capp --file ~/ test.ova
```

Field	Value
created_at	2015-12-02T06:56:53.000000
description	
fault	
id	d884e398-308b-4851-978f-4974e7eabcb3
image_ids	
is_protected	False
is_public	False
name	test_capp
status	Creating
type	OVF
updated_at	2015-12-02T06:56:55.000000
user_id	9a89e98b6b764a37a83ba44a27161352

Example 2 Create a cApp with Type OVF by using an OpenStack Client



```
root@atlas:~# openstack capp create --type ovf --name test_capp --file ~/ test.ova
```

Field	Value
created_at	2015-12-02T06:56:53.000000
description	
fault	
id	d884e398-308b-4851-978f-4974e7eabcb3
image_ids	
is_protected	False
is_public	False
name	test_capp
status	Creating
type	OVF
updated_at	2015-12-02T06:56:55.000000
user_id	9a89e98b6b764a37a83ba44a27161352

Example 3 Example Create a cApp with nested templates of Type HOT by Using an OVFT Client

```
root@atlas:~# openstack capp create --type ovf --name test_capp --file ~/ test.ova
```

Field	Value
created_at	2015-12-02T06:56:53.000000
description	
fault	
id	d884e398-308b-4851-978f-4974e7eabcb3
image_ids	
is_protected	False
is_public	False
name	test_capp
status	Creating
type	OVF
updated_at	2015-12-02T06:56:55.000000
user_id	9a89e98b6b764a37a83ba44a27161352

Example 4 Create a cApp with nested templates of Type HOT by Using an OpenStack Client

```
root@atlas:~# ovft capp-create --name test --file ~/test.yaml --type hot
```

Property	Value
created_at	2015-12-02T06:59:59.000000
description	
fault	
id	b0c50a44-6f89-485b-8d10-c9b1eda309f3
image_ids	
is_protected	False
is_public	False
name	test
status	Creating
type	HOT
updated_at	2015-12-02T06:59:59.000000
user_id	9a89e98b6b764a37a83ba44a27161352

Example 5 Create a cApp with Type HOT by Using an OVFT Client



```
root@atlas:~# openstack capp create --name test --file ~/test.yaml --type hot
```

Field	Value
created_at	2015-12-02T07:01:27.000000
description	
fault	
id	b2ccb378-ca2a-429f-9816-e0b9443d12d1
image_ids	
is_protected	False
is_public	False
name	test
status	Creating
type	HOT
updated_at	2015-12-02T07:01:27.000000
user_id	9a89e98b6b764a37a83ba44a27161352

Example 6 Create a cApp with Type HOT by Using an OpenStack Client

```
root@atlas:~# ovft capp-create --name test --file ~/test.yaml --type tosca
```

Property	Value
created_at	2015-12-02T07:03:02.000000
description	
fault	
id	a9301219-1d8c-441e-bbd3-8da7fc8ad077
image_ids	
is_protected	False
is_public	False
name	test
status	Creating
type	TOSCA
updated_at	2015-12-02T07:03:02.000000
user_id	9a89e98b6b764a37a83ba44a27161352

Example 7 Create a cApp with Type TOSCA by Using an OVFT Client

```
root@atlas:~# openstack capp create --name test --file ~/test.yaml --type tosca
```

Field	Value
created_at	2015-12-02T07:03:18.000000
description	
fault	
id	66616389-b434-4e73-9a3e-5a5d369f0a76
image_ids	
is_protected	False
is_public	False
name	test
status	Creating
type	TOSCA
updated_at	2015-12-02T07:03:18.000000
user_id	9a89e98b6b764a37a83ba44a27161352

Example 8 Create a cApp with Type TOSCA by Using an OpenStack Client

2.2 Create a cApp with All Option Fields

This section describes how to create a cApp with desc, is-public, and is-protected fields.



Syntax for OVFT client:

```
ovft capp-create --type <ovf/hot/tosca> ⇒  
--name <capp_name> --file <path_to_the_capp> ⇒  
--desc <capp_description> --is-public <true/false> ⇒  
--is-protected <true/false>
```

Syntax for OpenStack client:

```
openstack capp create --type <ovf/hot/tosca> ⇒  
--name <capp_name> --file <path_to_the_capp> ⇒  
--desc <capp_description> --is-public <true/false> ⇒  
--is-protected <true/false>
```

Examples:

```
root@atlas:~# ovft capp-create --name test --file test.ova --type ovf⇒  
--desc "Create test capp" --is-public true --is-protected true
```

Property	Value
created_at	2015-12-02T07:09:08.000000
description	Create test capp
fault	
id	f6256648-7574-42f4-897d-068484a20d30
image_ids	
is_protected	True
is_public	True
name	test
status	Creating
type	OVF
updated_at	2015-12-02T07:09:09.000000
user_id	9a89e98b6b764a37a83ba44a27161352

Example 9 Create a cApp with All Options by Using an OVFT Client



```
root@atlas:~# openstack capp create --name test --file test.ova --type ovf⇒
--desc "Create test capp" --is-public true --is-protected true
```

Field	Value
created_at	2015-12-02T07:09:55.000000
description	Create test capp
fault	
id	c8511b65-dd3c-429a-8687-ed21bc610168
image_ids	
is_protected	True
is_public	True
name	test
status	Creating
type	OVF
updated_at	2015-12-02T07:09:57.000000
user_id	9a89e98b6b764a37a83ba44a27161352

Example 10 Create a cApp with All Options by Using an OpenStack Client



3 cApp Delete

This section describes how to delete cApps.

Syntax for OVFT client:

```
ovft capp-delete <capp_name_or_capp_id>
```

Syntax for OpenStack client:

```
openstack capp delete <capp_name_or_capp_id>
```

Note: Do not use `capp_name` in these commands if there are multiple cApps available with the same name.

Examples:

```
root@atlas:~# ovft capp-delete c8511b65-dd3c-429a-8687-ed21bc610168
```

Example 11 Delete a cApp by cApp Name by Using an OVFT Client

```
root@atlas:~# openstack capp delete c8511b65-dd3c-429a-8687-ed21bc610168
```

Example 12 Delete a cApp by cApp Name by Using an OpenStack Client



4 cApp Update

This section describes how to update cApps.

Syntax for OVFT client:

```
ovft capp-update <existing_capp_name or existing_capp_id> --name <new_name>
```

Syntax for OpenStack client:

```
openstack capp update <existing_capp_name or existing_capp_id> --name <new_name>
```

Note: Do not use **existing_capp_name** in these commands if there are multiple cApps available with the same name.

The following arguments can be updated for an uploaded application:

- name <name>** The name of the cApp
- file <file>** A local file that contains disk cApp to be uploaded during the update. Alternatively, cApps can be passed to the client by using Standard Input (stdin).
- is-public {true,false}** Makes the cApp accessible to the public.
- is-protected {true,false}** Prevents the cApp from being deleted.
- desc <description>** Description of the cApp

Examples:

```
root@atlas:~# ovft capp-update c8511b65-dd3c-429a-8687-ed21bc610168 --name test1
```

Property	Value
created_at	2015-12-02T07:09:55.000000
description	Create test capp
fault	
id	c8511b65-dd3c-429a-8687-ed21bc610168
image_ids	b97a9ac5-9f04-49ce-9475-caa29702d574
is_protected	True
is_public	True
name	test1
status	active
type	OVF
updated_at	2015-12-02T07:15:40.000000
user_id	9a89e98b6b764a37a83ba44a27161352

Example 13 Update cApp by Using an OVFT Client



```
root@atlas:~# openstack capp update c8511b65-dd3c-429a-8687-ed21bc610168 --name test
```

Field	Value
created_at	2015-12-02T07:09:55.000000
description	Create test capp
fault	
id	c8511b65-dd3c-429a-8687-ed21bc610168
image_ids	b97a9ac5-9f04-49ce-9475-caa29702d574
is_protected	True
is_public	True
name	test
status	active
type	OVF
updated_at	2015-12-02T07:17:16.000000
user_id	9a89e98b6b764a37a83ba44a27161352

Example 14 Update cApp by Using an OpenStack Client



5 cApp Show

This section describes how to display the metadata of cApps.

Syntax for OVFT client:

```
ovft capp-show <capp_name_or_capp_id>
```

Syntax for OpenStack client:

```
openstack capp show <capp_name_or_capp_id>
```

Note: Do not use `capp_name` in these commands if there are multiple cApps available with the same name.

Examples:

```
root@atlas:~# ovft capp-show c8511b65-dd3c-429a-8687-ed21bc610168
```

Property	Value
created_at	2015-12-02T07:09:55.000000
description	Create test capp
fault	
id	c8511b65-dd3c-429a-8687-ed21bc610168
image_ids	b97a9ac5-9f04-49ce-9475-caa29702d574
is_protected	True
is_public	True
name	test
status	active
type	OVF
updated_at	2015-12-02T07:17:16.000000
user_id	9a89e98b6b764a37a83ba44a27161352

Example 15 Show cApp by Using an OVFT Client



```
root@atlas:~# openstack capp show c8511b65-dd3c-429a-8687-ed21bc610168
```

Field	Value
created_at	2015-12-02T07:09:55.000000
description	Create test capp
fault	
id	c8511b65-dd3c-429a-8687-ed21bc610168
image_ids	b97a9ac5-9f04-49ce-9475-caa29702d574
is_protected	True
is_public	True
name	test
status	active
type	OVF
updated_at	2015-12-02T07:17:16.000000
user_id	9a89e98b6b764a37a83ba44a27161352

Example 16 Show cApp by Using an OpenStack Client



6 cApp Show Template

This section describes how to display the stored HOT or TOSCA template of a cApp.

Syntax for OVFT client:

```
ovft capp-template-show <capp_name_or_capp_id>
```

Syntax for OpenStack client:

```
openstack capp template show <capp_name_or_capp_id>
```

Note: Do not use `capp_name` in these commands if there are multiple cApps available with the same name.

Examples:

```
root@atlas:~# ovft capp-template-show c8511b65-dd3c-429a-8687-ed21bc610168
description: stack template generated from OVF file
heat_template_version: '2013-05-23'
parameters:
  param_1:
    description: IP subnet
    label: BAT-OVF-ctrl1_100 subnet
    type: string
  param_2:
    description: Gateway
    label: BAT-OVF-ctrl1_100 gateway
    type: string
  param_3:
    description: Select fixedip within BAT-OVF-BGW-L3-3958_subnet
    label: Fixed IP defined for port vfr1_vnic2
    type: string
  param_4:
    description: Select fixedip within BAT-OVF-BGW-L3-3959_subnet
    label: Fixed IP defined for port vfr1_vnic3
    type: string
resources:
  BAT-A1-001:
    depends_on: delay_6
    properties:
      config_drive: 'True'
      flavor:
        get_resource: flavor_BAT-A1-001
      image: b97a9ac5-9f04-49ce-9475-caa29702d574
      metadata: {}
      name: BAT-A1-001
      networks:
        - port:
            get_resource: BAT-A1-001_existing_port_1
        - port:
            get_resource: port_46
        - port:
            get_resource: port_48
      scheduler_hints:
        group:
          get_resource: Affinity-1
      type: OS::Nova::Server
  Affinity-1:
    properties:
      name: Affinity-1
      policies:
        - AFFINITY
      type: OS::Nova::ServerGroup
  BAT-A1-001_existing_port_1:
    properties:
```



```
    name: BAT-A1-001_existing_port_1
    network_id: 4df3ff29-a31e-464a-943b-291c778e8881
    type: OS::Neutron::Port
BAT-A1-003:
  depends_on:
    - BAT-A2-001
    - BAT-A2-003
    - BAT-A2-005
    - BAT-A2-007
  properties:
    config_drive: 'True'
    flavor:
      get_resource: flavor_BAT-A1-003
    image: b97a9ac5-9f04-49ce-9475-caa29702d574
    metadata: {}
    name: BAT-A1-003
    networks:
      - port:
          get_resource: port_13
      - port:
          get_resource: port_15
    scheduler_hints:
      group:
        get_resource: Affinity-2
    type: OS::Nova::Server
Affinity-2:
  properties:
    name: Affinity-2
    policies:
      - AFFINITY
    type: OS::Nova::ServerGroup
```

Example 17 Display HOT Template of a cApp by Using an OVFT Client



```

root@atlas:~# openstack capp template show c8511b65-dd3c-429a-8687-ed21bc610168
description: stack template generated from OVF file
heat_template_version: '2013-05-23'
parameters:
  param_1:
    description: IP subnet
    label: BAT-OVF-ctrl_100 subnet
    type: string
  param_2:
    description: Gateway
    label: BAT-OVF-ctrl_100 gateway
    type: string
  param_3:
    description: Select fixedip within BAT-OVF-BGW-L3-3958_subnet
    label: Fixed IP defined for port vfr1_vnic2
    type: string
  param_4:
    description: Select fixedip within BAT-OVF-BGW-L3-3959_subnet
    label: Fixed IP defined for port vfr1_vnic3
    type: string
resources:
  BAT-A1-001:
    depends_on: delay_6
    properties:
      config_drive: 'True'
      flavor:
        get_resource: flavor_BAT-A1-001
      image: b97a9ac5-9f04-49ce-9475-caa29702d574
      metadata: {}
      name: BAT-A1-001
      networks:
        - port:
            get_resource: BAT-A1-001_existing_port_1
        - port:
            get_resource: port_46
        - port:
            get_resource: port_48
      scheduler_hints:
        group:
          get_resource: Affinity-1
      type: OS::Nova::Server
  Affinity-1:
    properties:
      name: Affinity-1
      policies:
        - AFFINITY
    type: OS::Nova::ServerGroup

```

Example 18 Display HOT Template of a cApp by Using an OpenStack Client



7 List Available cApps

This section describes how to list all the uploaded cApps.

Syntax for OVFT client:

ovft capp-list

Syntax for OpenStack client:

openstack capp list

Examples:

```
root@atlas:~# ovft capp-list
+-----+-----+-----+-----+-----+
| id | name | status | type | created_at |
+-----+-----+-----+-----+-----+
| c8511b65-dd3c-429a-8687-ed21bc610168 | test | active | OVF | 2015-12-02T07:09:55.000000 |
+-----+-----+-----+-----+-----+
```

Example 19 List Available cApps by Using an OVFT Client

```
root@atlas:~# openstack capp list
+-----+-----+-----+-----+
| id | name | status | created_at |
+-----+-----+-----+-----+
| 7a6478c2-18ff-4386-b07f-75f4c7c1b522 | tests | None | 2015-12-02T07:21:56.000000 |
+-----+-----+-----+-----+
```

Example 20 List Available cApps by Using an OpenStack Client



8 List Files for a cApp

This section describes how to list the files for an OVF type cApp.

Syntax for OVFT client:

```
ovft capp-file-list <capp_name_or_capp_id>
```

Syntax for OpenStack client:

```
openstack capp file list <capp_name_or_capp_id>
```

Note: Do not use `capp_name` in these commands if there are multiple cApps available with the same name.

Examples:

file_id	file_name
76e9192f-50a1-4ea9-9c38-573b09309545	user-data
8ed1cbbb-147d-41a3-9276-2b19000eb9fb	new/two_nova.yaml
c81d8886-6b6e-4a54-9d88-6cd40ab6b532	new/three_nova.yaml
f8cd2758-58b8-4068-94f1-c1ebba76dfce	new/tripled_nest.yaml

Example 21 List Files for a cApp by Using an OVFT Client

file_id	file_name
76e9192f-50a1-4ea9-9c38-573b09309545	user-data
8ed1cbbb-147d-41a3-9276-2b19000eb9fb	new/two_nova.yaml
c81d8886-6b6e-4a54-9d88-6cd40ab6b532	new/three_nova.yaml
f8cd2758-58b8-4068-94f1-c1ebba76dfce	new/tripled_nest.yaml

Example 22 List Files for a cApp by Using an OpenStack Client



9 Update File Content

This section describes how to update the content of the file for a cApp.

Note: Modifying the referenced file names in files is not allowed.

Do not use `file_name` in these commands.

Syntax for OVFT client:

```
ovft capp-file-update [--input <input_file_path>] <file_id>
```

Syntax for OpenStack client:

```
openstack capp file update [--input <input_file_path>] =>  
<file_id>
```

Examples:

```
root@atlas:/home/atlasadm# ovft capp-file-show bc320c1c-91df-4bd3-8e15-a7a1edcd3799
```

Property	Value
file_content	heat_template_version: 2015-04-30 parameters: image_name: type: string description: Name of a image resources: server: type: OS::Nova::Server properties: flavor: m1.small image: {get_param: image_name} networks: - network: provider_51 user_data: {get_file: ../user-data} my_server_new: type: two_nova.yaml properties: net_name: provider_51
file_id	bc320c1c-91df-4bd3-8e15-a7a1edcd3799
file_name	new/tripled_nest.yaml

Example 23 File Content Before Update



```
root@atlas:~#cat new/nest.yaml
heat_template_version: 2015-04-30

parameters:
  image_name:
    type: string
    description: Name of a image

resources:
  server:
    type: OS::Nova::Server
    properties:
      flavor: m1.medium
      image: {get_param: image_name}
      networks:
        - network: provider_51
      user_data: {get_file: ../user-data}
  my_server_new:
    type: two_nova.yaml
    properties:
      net_name: provider_51
```

Example 24 Input File Content

```
root@atlas:~# ovft capp-file-update --input new/nest.yaml 1e0d698f-3e7f-4d92-ae36-8f09d1b4e3f4
```

Property	Value
file_content	<pre>heat_template_version: 2015-04-30 parameters: image_name: type: string description: Name of a image resources: server: type: OS::Nova::Server properties: flavor: m1.medium image: {get_param: image_name} networks: - network: provider_51 user_data: {get_file: ../user-data} my_server_new: type: two_nova.yaml properties: net_name: provider_51</pre>
file_id	1e0d698f-3e7f-4d92-ae36-8f09d1b4e3f4
file_name	new/tripled_nest.yaml

Example 25 Update File Content by Using an OVFT Client



```
root@atlas:~# openstack capp file update --input new/nest.yaml 1e0d698f-3e7f-4d92-ae36-8f09d1b4e3f4
```

Property	Value
file_content	heat_template_version: 2015-04-30 parameters: image_name: type: string description: Name of a image resources: server: type: OS::Nova::Server properties: flavor: m1.medium image: {get_param: image_name} networks: - network: provider_51 user_data: {get_file: ../user-data} my_server_new: type: two_nova.yaml properties: net_name: provider_51
file_id	1e0d698f-3e7f-4d92-ae36-8f09d1b4e3f4
file_name	new/tripled_nest.yaml

Example 26 Update File Content by Using an OpenStack Client



10 Show File Content

This section describes how to display the content of the file for an OVF type cApp.

Syntax for OVFT client:

```
ovft capp-file-show <capp_name_or_capp_id>
```

Syntax for OpenStack client:

```
openstack capp file show <capp_name_or_capp_id>
```

Note: Do not use `capp_name` in these commands if there are multiple cApps available with the same name.

Examples:

```
root@atlas:~# ovft capp-file-show 67a0c543-72e7-4951-ba36-419963797f06
```

Property	Value
file_content	env file example
file_id	67a0c543-72e7-4951-ba36-419963797f06
file_name	config2.xml

Example 27 Show File Content by Using an OVFT Client

```
root@atlas:~# openstack capp file show 67a0c543-72e7-4951-ba36-419963797f06
```

Field	Value
file_content	env file example
file_id	67a0c543-72e7-4951-ba36-419963797f06
file_name	config2.xml

Example 28 Show File Content by Using an OpenStack Client



11 cApp Export

This section describes how to export cApps from the database to the Atlas file system.

Syntax for OVFT client:

```
ovft capp-export --file <local_file_for_saving_downloaded_capp_data> <capp_name or capp_id>
```

Syntax for OpenStack client:

```
openstack capp export --file <local_file_for_saving_downloaded_capp_data> <capp_name or capp_id>
```

Note: Do not use **capp_name** in these commands if there are multiple cApps available with the same name.

Examples:

```
root@atlas:~# ovft capp-export --file ~/check c8511b65-dd3c-429a-8687-ed21bc610168
```

Example 29 Export a cApp by Using an OVFT Client

```
root@atlas:~# openstack capp export --file ~/check c8511b65-dd3c-429a-8687-ed21bc610168
```

Example 30 Export a cApp by Using an OpenStack Client



12 Application Scaling

This section describes how to manually scale resources of a running stack.

12.1 Stack Scale Out

Stack-scale (--method out) creates all the resources mentioned in the scale out JavaScript Object Notation (JSON) file.

The Scale Out JSON file contains the following sections:

scaling_groups	Contains the details of the resources.
resources	Contains the list of resources to scale.
source	Name of the server you want to scale
target	Name of the server with which the source server gets scaled
network(Opt)	Mac address or IP address for a particular network: <ul style="list-style-type: none">• name Name of network• mac_address(Opt) Mac address for the target VM• ip_address(Opt) Fixed IP address for the target VM
personality(Opt)	Personality section for the target VM

Note: (Opt) in the section names stands for (Optional). The Mac Address, Fixed IP address, and Personality sections are optional.

The following example shows the content of the Scale Out JSON file.



```

root@atlas:~# cat scale_out_fixed_ip_mac.json
{ "scaling_groups": {
  "resources": [
    {
      "source" : "vm1",
      "target" : "vm1_scaled",
      "networks" : [
        {
          "name" : "N1",
          "mac_address" : "44:40:44:40:44:33",
          "ip_address" : "10.0.0.18"
        }
      ],
      "personality": [
        {
          "contents": "Something",
          "path": "/etc/script.sh"
        }
      ]
    }
  ]
}
}

```

Example 31 The Scale Out JSON File

Syntax for OVFT client:

```

ovft stack-scale --method out --file <scale_out_json_file> <stack_name =>
or stack_id>

```

Syntax for OpenStack client:

```

openstack stack scale --method out --file <scale_out_json_file> <stack_name =>
or stack_id>

```

Note: Do not use `capp_name` in these commands if there are multiple cApps available with the same name.

Examples:

```

root@atlas:~# ovft stack-scale --file scale_out.json --method out =>
scale_out_stack

```

```

+-----+-----+
| Property | Value      |
+-----+-----+
| resources | vm1_scaled |
+-----+-----+

```

Example 32 Scale Out a Stack by Using an OVFT Client

```

root@atlas:~# openstack stack scale --file scale_out.json --method out scale_out_stack

```

```

+-----+-----+
| Field   | Value      |
+-----+-----+
| resources | vm1_scaled |
+-----+-----+

```

Example 33 Scale Out a Stack by Using an OpenStack Client



```
atlasadm@atlas:~$ heat resource-list 6866f864-3231-49ff-bdc4-99db84d69b3c
```

resource_name	physical_resource_id
N1	7a8705b5-4e33-4155-9f03-38b07ad4e888
N1_subnet	68b4dd6f-995d-4e5d-9653-53b2a1d0b533
flavor_vm1	cdc480c6-68df-4411-96ed-3a68622012f7
vm1	ba67ef98-a0e5-4565-bab3-69f30be1f86f
vm1_scaled	bde6733b-677a-461b-a558-37ecf119943e

resource_type	resource_status	updated_time
OS::Neutron::Net	CREATE_COMPLETE	2015-12-02T07:49:04Z
OS::Neutron::Subnet	CREATE_COMPLETE	2015-12-02T07:49:04Z
OS::Nova::Flavor	CREATE_COMPLETE	2015-12-02T07:49:04Z
OS::Nova::Server	CREATE_COMPLETE	2015-12-02T07:49:04Z
OS::Heat::OnDemandScalingGroup	CREATE_COMPLETE	2015-12-02T07:49:54Z

Example 34 Stack Resource List

12.2 Stack Scale In

Stack-scale (--method in) deletes all the scaled resources associated to a stack mentioned in the scale in JSON file.

The Scale In JSON file contains the following section:

groups List of groups you want to scale in

The following example shows the content of the Scale In JSON file.

```
root@atlas:~# cat scale_in.json
{
    "groups" : [
        "vm1_scaled"
    ]
}
```

Example 35 The Scale In JSON File

Syntax for OVFT client:

```
ovft stack-scale --method in =>
--file <scale_in_json_file> <stack_name>
```

Syntax for OpenStack client:

```
openstack stack scale --method in =>
--file <scale_in_json_file> <stack_name>
```

Examples:



```
root@atlas:~# ovft stack-scale --file scale_in.json --method in scale_out_stack
```

```
+-----+-----+
| Property | Value |
+-----+-----+
| scaled_info | OK |
+-----+-----+
```

Example 36 Scale In a Stack by Using an OVFT Client

```
root@atlas:~# openstack stack scale --file scale_in.json --method in scale_out_stack
```

```
+-----+-----+
| Field | Value |
+-----+-----+
| scaled_info | OK |
+-----+-----+
```

Example 37 Scale In a Stack by Using an OpenStack Client

12.3 Stack Scaling List

The stack-scaling-list lists all the scaling groups associated to a stack.

Syntax for OVFT client:

```
ovft stack-scaling-list <stack_name>
```

Syntax for OpenStack client:

```
openstack stack scaling list <stack_name>
```

Examples:

```
root@atlas:~# ovft stack-scaling-list scale_out_stack
```

```
+-----+-----+
| Property | Value |
+-----+-----+
| scaling groups | vm1_scaled |
| servers | vm1 |
+-----+-----+
```

Example 38 Stack Scaling List by Using an OVFT Client

```
root@atlas:~# openstack stack scaling list scale_out_stack
```

```
+-----+-----+
| Field | Value |
+-----+-----+
| scaling groups | vm1_scaled |
| servers | vm1 |
+-----+-----+
```

Example 39 Stack Scaling List by Using an OpenStack Client



13 cApp Personalize

cApp Personalize allows the user to customize resource properties in a cApp template, which can either be stored into an output HOT template or uploaded to the database directly.

13.1 Adding Properties From CLI

Properties can be added through cApp Personalize using the specified environment file input.

cApp Personalize supports the following properties:

- User data
- Metadata
- File injection

Note: As the maximum body size for HTTP requests is set to 112 KB in the default settings of the Nova API, larger environment files cannot be injected, even if the `injected_file_content_bytes` quota value is changed for the specific tenant. To increase the default value, set `osapi_max_request_body_size` in `/etc/nova/nova.conf` on all controllers and restart `nova-api`.

- Availability zone
- `extra_specs` for flavor, or use of existing flavor

The above mentioned properties can be added into an existing cApp, to generate a new HOT template that can be used to launch stack with the added properties. The added properties can be updated in the database by using the `-s` option.

Syntax for OVFT client:

```
ovft capp-template-personalize -e <input_env_file> =>
-o <output_file_name> [-s] <capp_id>
```

The use of `-s` is optional. Only use `-s` if the changes need to be reflected in the database as well.

```
root@atlas:~# ovft capp-template-personalize -e personalize-env.yaml -o output.yaml capp_id
```

Example 40 cApp Personalize Command for Adding Properties



```

root@atlas:~# cat personalize-env.yaml
resource_registry:
  resources:
    Controller_node1:
      properties:
        availability_zone: "nova"
        personality:
          /home/files/enivronmentfile.xml:
            get_file: /tmp/env.xml
      user_data:
        str_replace:
          template: |
            #!/bin/bash
            chmod +x $path_to_setup_file
            $path_to_setup_file --interface
            $path_to_setup_file --copy
            $path_to_setup_file --setup_remote
          params:
            $path_to_setup_file: /home/atlasadm/user_data
      metadata:
        app_ip: IP of the application
      type: OS::Nova::Server
    flavor_Controller_node1:
      properties:
        extra_specs:
          "quota:disk_read_bytes_sec": "10240000"
      type: OS::Nova::Flavor

```

Example 41 Input environment File Format

The output (output.yaml) will be generated with the properties mentioned in Example 41 updated/added:

```

root@atlas:~# cat output.yaml
description: stack template generated from OVF file
heat_template_version: '2013-05-23'
parameters:
  param_1: {description: Segmentation ID of network, label: demo segmentation ID,
    type: string}
  param_2: {description: IP subnet, label: demo subnet, type: string}
  param_3: {description: Gateway, label: demo gateway, type: string}
resources:
  Controller_node1:
    properties:
      availability_zone: nova
      config_drive: 'True'
      flavor: {get_resource: flavor_Controller_node1}
      image: 22603a05-9efe-4d9f-a92c-53a8bd49ff06
      metadata: {app_ip: IP of the application}
      name: Controller node1
      networks:
        - network: {get_resource: demo}
      personality: {/home/files/enivronmentfile.xml: 'This is environment
        ', setup/cfg.xml: "<instance name=\"cirros\">\n <image name=\"cirros_img\" \
>cirros_img</image>\n <flavor name=\"m1.tiny\">m1.tiny</flavor>\n <property \
\ key='username' value='$username'/>\n</instance>\n", setup/resources.xml: "<instance \
\ name=\"cirros\">\n <image name=\"cirros_img\">cirros_img</image>\n <flavor \
\ name=\"m1.tiny\">m1.tiny</flavor>\n <property key='username' value='$username'/>\n \
</instance>\n"}
      user_data:
        str_replace:
          params: {$path_to_setup_file: /home/atlasadm/user_data}
          template: '#!/bin/bash

            chmod +x $path_to_setup_file

            $path_to_setup_file --interface

            $path_to_setup_file --copy

            $path_to_setup_file --setup_remote

```




```

    type: OS::Nova::Server
Payload_node1:
  properties:
    config_drive: 'True'
    flavor: {get_resource: flavor_Payload_node1}
    image: 3f9c0250-f741-435a-b318-0a8e8fa3e6b0
    metadata: {}
    name: Payload node1
    networks:
      - network: {get_resource: demo}
  type: OS::Nova::Server
demo:
  properties:
    name: demo
    value_specs:
      provider:network_type: vlan
      provider:physical_network: default
      provider:segmentation_id: {get_param: param_1}
      router:external: true
  type: OS::Neutron::Net
demo_subnet:
  properties:
    cidr: {get_param: param_2}
    enable_dhcp: true
    gateway_ip: {get_param: param_3}
    ip_version: '4'
    name: demo_subnet
    network_id: {get_resource: demo}
  type: OS::Neutron::Subnet
flavor_Controller_node1:
  properties:
    disk: 1
    extra_specs: {'quota:disk_read_bytes_sec': '10240000'}
    ram: 1024
    vcpus: 1
  type: OS::Nova::Flavor
flavor_Payload_node1:
  properties: {disk: 1, ram: 1024, vcpus: 1}
  type: OS::Nova::Flavor

```

Example 42 Output Template Generated

13.2 cApp Personalize for Editing the Main Template

In addition to the properties in Section 13.1 on page 27, further minor modifications can be done to the main template. By providing the new updated template as input to the cApp personalize command, the input template will replace the stored cApp template.

Note: Modifying the referenced file names in the main template is not allowed.

Syntax for OVFT client:

```
ovft capp-template-personalize --input <input_file> =>
<capp_id>
```

```
root@atlas:~# ovft capp-template-personalize --input update.yaml capp_id
```

Example 43 Deployment Wizard Command for Updating the Main Template with Minor Changes



```
root@atlas:~$ cat update.yaml
heat_template_version: 2015-04-30
resources:
  server111:
    type: OS::Nova::Server
    properties:
      flavor: m1.medium
      image: 620b69d2-02cc-412c-a717-6a277215a015
      networks:
        - network: provider_51
      user_data: {get_file: user-data}
  my_server:
    type: new/tripled_nest.yaml
    properties:
      image_name: cirros-0.3.2-x86_64-disk
  my_server111:
    type: new/three_nova.yaml
    properties:
      net_name: provider_50
```

Example 44 Input File with Updates

```
root@atlas:~$ ovft capp-template-show sample
heat_template_version: 2015-04-30
resources:
  server111:
    type: OS::Nova::Server
    properties:
      flavor: m1.small
      image: 620b69d2-02cc-412c-a717-6a277215a015
      networks:
        - network: provider_51
      user_data: {get_file: user-data}
  my_server:
    type: new/tripled_nest.yaml
    properties:
      image_name: cirros-0.3.2-x86_64-disk
  my_server111:
    type: new/three_nova.yaml
    properties:
      net_name: provider_50
```

Example 45 Main Template Before Update

```
root@atlas:~$ ovft capp-template-show sample
heat_template_version: 2015-04-30
resources:
  server111:
    type: OS::Nova::Server
    properties:
      flavor: m1.medium
      image: 620b69d2-02cc-412c-a717-6a277215a015
      networks:
        - network: provider_51
      user_data: {get_file: user-data}
  my_server:
    type: new/tripled_nest.yaml
    properties:
      image_name: cirros-0.3.2-x86_64-disk
  my_server111:
    type: new/three_nova.yaml
    properties:
      net_name: provider_50
```

Example 46 Main Template After Update