OpenNebula OneScape 5.10 Documentation

Release 5.10.0

OpenNebula Systems

Mar 24, 2020

CONTENTS

OneScape is commercial enterprise extensions to the open-source OpenNebula which aims to provide means to simplify maintenance, management and upgrade of the OpenNebula deployments. It's an innovative and independent component with an ambition to evolve into the central control, maintenance and monitoring point of the installation. It consists of a growing set of modules which provide essential functionality for the enterprise environments or improve the experience for the specific part of maintenance workflow.

Current features are limited to managing *configuration files* and generating *diagnostic bundle*.

Important: You need to have valid **support subscription** to get access this component. Follow the available options of Enteprise Support Subscriptions.

CHAPTER

ONE

RELEASE NOTES

1.1 What's New in 5.10

OneScape 5.10 is the very first release of the component and focuses on improving the OpenNebula **upgrade experience of the configuration files**.

1.1.1 Configuration Module

OpenNebula upgrade process was always recognized as much easier when compared to similar products. Software components are upgraded by standard operating system package managers (`yum` or `apt`), and the cloud state is easily migrated by OpenNebula database management tool (`onedb`). The only step which required special attention and expertise was the migration of custom changes in the configuration files to the new version. Responsible Cloud Infrastructure Administrator had to keep track of custom changes (or be able to identify them) and correctly migrate them to the new version of configuration files. From all the required upgrade steps, the migration of changes in configuration files was the most unpopular one as it was heavily (and usually) manual process.

OneScape introduces the new command-line tool `onecfg` for the **automatic upgrade of configuration files** for new OpenNebula versions while preserving the customizations. Supported upgrade paths are from OpenNebula 5.4.x, 5.6.x or 5.8.x directly to 5.10.2 (or newer).

Read more in the *Configuration Module*.

1.1.2 Support Module

Tools to easily generate support diagnostic bundles for the support cases are part of a new *Support Module*.

1.2 Platform Notes

This is the list of the individual platform components that have been through the complete OpenNebula Quality Assurance and Certification Process.

1.2.1 Certified Components Version

OpenNebula

Component	Version
Front-end	5.10 (from release 5.10.2)
Upgrades From Versions	5.4.0 - 5.8.5

Operating Systems

Component	Version
Red Hat Enterprise Linux	7, 8
CentOS	7, 8
Ubuntu Server	16.04 (LTS), 18.04 (LTS)
Debian	9, 10

CHAPTER

TWO

INSTALLATION

2.1 Repository Setup

Distribution of the OneScape is via the operating system packages for the *supported platforms*. First, you need to configure your host to have access to dedicated authenticated customer repositories, then you can install or update OneScape packages.

Access to the software repositories with OneScape is authenticated by the customer's user name and token. Repository definitions below contain the placeholders `${AUTH}` where customer's credentials should be put. To simplify the process and to avoid dealing with the repository configurations manually, export your credentials following way before running the next commands.

```
# export AUTH='user:token'
```

2.1.1 CentOS and RHEL

To add OneScape repository execute the following as root:

CentOS/RHEL 7

```
# cat << EOT > /etc/yum.repos.d/onescape.repo
[onescape]
name=onescape
baseurl=https://$AUTH@enterprise.opennebula.io/repo/onescape/5.10/CentOS/7/$basearch
enabled=1
gpgkey=https://downloads.opennebula.io/repo/repo.key
gpgcheck=1
repo_gpgcheck=1
EOT
```

CentOS/RHEL 8

```
# cat << EOT > /etc/yum.repos.d/onescape.repo
[onescape]
name=onescape
baseurl=https://$AUTH@enterprise.opennebula.io/repo/onescape/5.10/CentOS/8/$basearch
enabled=1
gpgkey=https://downloads.opennebula.io/repo/repo.key
gpgcheck=1
repo_gpgcheck=1
EOT
```

2.1.2 Debian and Ubuntu

To add OneScape repository on Debian/Ubuntu execute as root:

```
# wget -q -O- https://downloads.opennebula.io/repo/repo.key | apt-key add -
```

Debian 9

```
# echo "deb https://$AUTH@enterprise.opennebula.io/repo/onescape/5.10/Debian/9 stable onescape" > /et
```

Debian 10

```
# echo "deb https://$AUTH@enterprise.opennebula.io/repo/onescape/5.10/Debian/10 stable onescape" > /e
```

Ubuntu 16.04

```
# echo "deb https://$AUTH@enterprise.opennebula.io/repo/onescape/5.10/Ubuntu/16.04 stable onescape" >
```

Ubuntu 18.04

```
# echo "deb https://$AUTH@enterprise.opennebula.io/repo/onescape/5.10/Ubuntu/18.04 stable onescape" >
```

2.2 Installation

To install OneScape, ensure you have configured access to the *software repository* with OneScape and execute one of the following commands below based on your operating system.

2.2.1 CentOS and RHEL

```
$ sudo yum install onescape
```

2.2.2 Debian and Ubuntu

```
$ sudo apt-get install onescape
```

2.3 Upgrade

To upgrade existing OneScape installation, execute one of the following commands below based on your operating system.

2.3.1 CentOS and RHEL

```
$ sudo yum update onescape
```

2.3.2 Debian and Ubuntu

```
$ sudo apt-get install onescape
```

CHAPTER

THREE

CONFIGURATION MANAGEMENT MODULE

3.1 Overview

The aim of the **Configuration Module** is to provide the means to manage and interact with configuration files. It provides the following functionality:

- Upgrade your configuration files for the new OpenNebula version.
- Check the versions status of the current installation.
- Identify custom changes made to the configuration files.
- Validate configuration files.

All the functionality is available through the single command-line tool `onecfg`.

3.1.1 How Should I Read This Chapter

In this chapter, you will find all the information about how to manage your configuration files. You also will see some information about how to deal with possible conflicts you might experience during the upgrade process.

3.2 Basic Usage

The configuration module is controlled via `onecfg` CLI tool. This section covers subcommands provided by the tool:

- *status* - Versions status
- *init* - Initialize version management state
- *validate* - Validate current configuration files
- *diff* - Identify changes in configuration files
- *upgrade* - Upgrade configuration files to a new version

Important: This command must be always run under privileged user `root` directly or via `sudo`. For example:

```
$ sudo onecfg status
```

The tool comes with help for each subcommand and command-line option. Simple run without any parameter or a run with the parameter `--help` prints the brief documentation (e.g., `onecfg status --help`).

3.2.1 Status

The `status` subcommand provides an overview of the OpenNebula installation. It shows:

- Current OpenNebula version.
- Current configuration files version.
- Backup from previous OpenNebula version to process.
- Available updates with the corresponding migrators.

Note: If status subcommand fails on an **unknown** configuration version, check the section about *init* subcommand below.

Example:

```
# onecfg status
--- Versions -----------------
OpenNebula: 5.10.1
Config:     5.6.0

--- Backup to Process ----------------------
Snapshot:    /var/lib/one/backups/config/backup
(will be used as one-shot source for next update)

--- Available updates --------
New config: 5.10.0
- from 5.6.0 to 5.8.0 (YAML,Ruby)
- from 5.8.0 to 5.10.0 (YAML,Ruby)
```

Important: **OpenNebula version** and **Configuration version** are tracked independently, but both versions are closely related and must be from the same X.Y release (i.e., OpenNebula 5.10.Z must have a configuration on version 5.10.Z). Minor configuration releases X.Y.Z are tight to the OpenNebula version for which a significant update has happened. Usually, configuration version **remains on the same version for all OpenNebula releases** within the same X.Y release (i.e., configuration version 5.8.0 is valid for all OpenNebula from 5.8.0 to 5.8.5 releases).

Backup to Process is a one-shot backup that needs to be processed. It's created automatically by OpenNebula packages (since 5.10.2) during the upgrade and contains a backup of all configuration files from the previous version. Content of the backup is taken, upgraded for current OpenNebula version and placed into production directories (`/etc/one/` and `/var/lib/one/remotes/etc`). Any existing content will be replaced there.

Example of status without available updates:

```
# onecfg status
--- Versions -----------------------------
OpenNebula:  5.10.2
Config:      5.10.0

--- Available Configuration Updates -------
No updates available.
```

Exit codes

Based on different status, the command ends with the following exit codes:

- **0** - No update available.

- **1** - Updates available.
- **255** - Unspecified error (e.g., unknown versions)

3.2.2 Init

For clean new installations, the `init` subcommand initializes the configuration management state based on the currently installed OpenNebula version.

Parameters:

Parameter	Description	Mandatory
`--force`	Force (re)initialization	NO
`--to` VERSION	Configuration version override (default: current OpenNebula version)	NO

Examples:

```
# onecfg init
INFO  : Initialized on version 5.10.0

# onecfg init
ANY   : Already initialized
```

You can also force configuration reinitialization based on detected OpenNebula version:

```
# onecfg init --force
INFO  : Initialized on version 5.10.0
```

Or, force reinitialization on own provided version:

```
# onecfg init --force --to 5.8.0
INFO  : Initialized on version 5.8.0
```

Note: Version state is stored in configuration file `/etc/onescape/config.yaml`. You **shouldn't modify this file directly**, as it might result in unpredictable behavior.

Example

Initialization is necessary when the OneScape is not sure about the version of current configuration files. When running `onecfg status` in the uninitialized environment, you might get the following error:

```
# onecfg status
--- Versions -------------------------------
OpenNebula:  5.8.0
Config:      unknown
ERROR: Unknown config version
```

If you are sure the configuration files are current for the OpenNebula version you have (i.e., 5.8.0 in the example above), you can initialize the version management by using OpenNebula version (e.g., `onecfg init`). Or, by explicitly providing the version configuration files match (e.g., `onecfg init --to 5.6.0`).

In both cases, after the initialization, the configuration version should be known:

```
# onecfg status
--- Versions -------------------------------
OpenNebula:  5.8.0
Config:      5.8.0
```

```
--- Available Configuration Updates -------
No updates available.
```

3.2.3 Validate

The `validate` subcommand checks that all known *configuration files* can be parsed.

Parameters:

Parameter	Description	Mandatory
`--prefix` PATH	Root location prefix (default: `/`)	NO

Without any parameter provided, it validates and returns only problematic files:

```
# onecfg validate
ERROR : Unable to process file '/etc/one/oned.conf' - Failed to parse file
```

When running in verbose mode with `--verbose`, it writes all checked files:

```
# onecfg validate --verbose
INFO  : File '/etc/one/vcenter_driver.default' - OK
INFO  : File '/etc/one/ec2_driver.default' - OK
INFO  : File '/etc/one/az_driver.default' - OK
INFO  : File '/etc/one/auth/ldap_auth.conf' - OK
INFO  : File '/etc/one/auth/server_x509_auth.conf' - OK
...
```

Note: You can also validate files inside a dedicated directory instead of a system-wide installation location using the option `--prefix`. Directory structure inside the prefix **must follow the structure on real locations** (e.g., for real `/etc/one` there must be `$PREFIX/etc/one`).

```
# onecfg validate --prefix /tmp/ONE --verbose
INFO  : File '/tmp/ONE/etc/one/vcenter_driver.default' - OK
INFO  : File '/tmp/ONE/etc/one/ec2_driver.default' - OK
INFO  : File '/tmp/ONE/etc/one/az_driver.default' - OK
INFO  : File '/tmp/ONE/etc/one/auth/ldap_auth.conf' - OK
INFO  : File '/tmp/ONE/etc/one/auth/server_x509_auth.conf' - OK
...
```

Exit codes

- **0** - all files are OK
- **1** - error when processing some file

3.2.4 Diff

Similar to the validation functionality above, the `diff` subcommand reads all *configuration files* and identifies changes that were done by the user when compared to based configuration files. It doesn't do any changes in the files, only reads and compares them.

Parameters:

Parameter	Description	Mandatory
`--prefix` PATH	Root location prefix (default: `/`)	NO

The command prints only files with changes. Unchanged files are not included in the output. Each individual change description is printed on a separate line with following syntax:

- `ins PATH = VALUE` - inserted new parameter on `PATH` with `VALUE`
- `set PATH = VALUE` - existing attribute on `PATH` was changed with different `VALUE`
- `rm PATH (= VALUE)` - deleted parameter on `PATH` (optionally specifying the removed `VALUE`)

Example:

```
# onecfg diff
/etc/one/cli/onegroup.yaml
- ins ID/adjust = true
- set NAME/size = 15
- ins NAME/expand = true

/etc/one/cli/onehost.yaml
- ins ID/adjust = true
- ins NAME/expand = true
- set CLUSTER/size = 10
- set STAT/size = 4

/etc/one/cli/oneimage.yaml
- ins ID/adjust = true
- set USER/size = 8
- set GROUP/size = 8
- ins NAME/expand = true

/etc/one/oned.conf
- set DEFAULT_DEVICE_PREFIX = "ŝd"
- set VM_MAD[NAME = '"vcenter"']/ARGUMENTS = "¨-p -t 15 -r 0 -s sh vcenter"
- rm  VM_MAD[NAME = '"vcenter"']/DEFAULT = "v̈mm_exec/vmm_exec_vcenter.conf"
- ins HM_MAD/ARGUMENTS = "¨-p 2101 -l 2102 -b 127.0.0.1"
- ins VM_RESTRICTED_ATTR = "ÑIC/FILTER"
...
```

How to read the output? Let's go through few examples for `/etc/one/cli/onegroup.yaml` above:

- `ins ID/adjust = true` - new key `adjust` with value `true` was added into `ID` section
- `set NAME/size = 15` - value for existing key `size` inside section `NAME` was changed to `15`

3.2.5 Upgrade

The `upgrade` subcommand makes all the changes in configuration files to update content from one version to another. It mainly does the following steps:

- Detect if an upgrade is necessary (or, at least if one-shot backup should be processed)
- Backup existing configuration files
- Apply upgrades (run migrators)
- Copy upgraded files back

Important: Upgrade operation is always done on a copy of your production configuration files in the temporary directory. If anything fails during the upgrade process, it doesn't affect the real files. When the upgrade is successfully done for all files and for all intermediate versions, the new state is copied back to production locations. In case of serious failure during the final copy back, there should be a backup stored in `/var/lib/one/backups/config/` for manual restore.

Note: You can first test the dry upgrade with `--noop`, which doesn't change real production files. It skips the final copy back phase.

Important: Upgrade operation detects changed values and preserves their content. Using patch mode **replace** described in *Troubleshooting*, the user can request to replace changed values with default ones for which **new default appears in the newer version**.

Parameters:

Parameter	Description	Mandatory
`--from` VERSION	Old configuration version (default: current)	NO
`--to` VERSION	New configuration version (default: autodetected from OpenNebula)	NO
`--noop`	Runs upgrade without changing system state	NO
`--unprivileged`	Skip privileged operations (e.g., `chown`) - only for testing	NO
`--patch-modes` MODES	Patch modes per file and version	NO
`--recreate`	Recreate deleted files that would be changed	NO
`--prefix` PATH	Root location prefix (default: `/`)	NO
`--read-from` PATH	Backup directory to take as source of current state (instead of production directories)	NO

In most cases, the upgrade from one version to another will be as easy as simply run of command `onecfg upgrade` without any extra parameters. It'll upgrade based on internal configuration version tracking and currently installed OpenNebula. For example:

```
# onecfg upgrade
ANY   : Backup stored in '/tmp/onescape/backups/2019-12-16_13:10:16_18130'
ANY   : Configuration updated to 5.10.0
```

Important: The upgrade process tries to apply changes from newer versions to your current configuration files (i.e., diff/patch approach modified for each different configuration file type). If the configuration files have been heavily modified, the upgrade might easily fail. The dedicated section describes how to *deal with conflicts* during the upgrade (patching) process.

If there is no upgrade available, the tool will not do anything:

```
# onecfg upgrade
ANY   : No updates available
```

To see the files changed during the upgrade, run the command in verbose mode via `--verbose` parameter. For example:

```
# onecfg upgrade --verbose
INFO  : Checking updates from 5.8.0 to 5.10.0
ANY   : Backup stored in '/tmp/onescape/backups/2019-12-12_15:14:39_18278'
INFO  : Updating from 5.8.0 to 5.10.0
INFO  : Incremental update from 5.8.0 to 5.10.0
INFO  : Update file '/etc/one/vcenter_driver.default'
INFO  : Skip file '/etc/one/cli/oneprovision.yaml' - missing
INFO  : Update file '/etc/one/cli/onegroup.yaml'
INFO  : Update file '/etc/one/cli/onehost.yaml'
INFO  : Update file '/etc/one/cli/oneimage.yaml'
...
```

Versions Override

It can be useful to control the upgrade process by providing custom source configuration version (`--from VERSION`), target configuration version (`--to VERSION`), or both configuration versions in cases when some version is not known or when user wants to have control over the process when upgrading over multiple major versions.

The example below demonstrates step-by-step manual upgrade with versions enforcing (verbose output was filtered):

```
# onecfg upgrade --verbose --from 5.4.0 --to 5.6.0
INFO  : Checking updates from 5.4.0 to 5.6.0
ANY   : Backup stored in '/tmp/onescape/backups/2019-12-17_18:08:05_28564'
INFO  : Updating from 5.4.0 to 5.6.0
INFO  : Incremental update from 5.4.0 to 5.4.1
INFO  : Incremental update from 5.4.1 to 5.4.2
INFO  : Incremental update from 5.4.2 to 5.4.6
INFO  : Incremental update from 5.4.6 to 5.6.0
ANY   : Configuration updated to 5.6.0

# onecfg upgrade --verbose --to 5.8.0
INFO  : Checking updates from 5.6.0 to 5.8.0
ANY   : Backup stored in '/tmp/onescape/backups/2019-12-17_18:10:18_29087'
INFO  : Updating from 5.6.0 to 5.8.0
INFO  : Incremental update from 5.6.0 to 5.8.0
ANY   : Configuration updated to 5.8.0

# onecfg upgrade --verbose
INFO  : Checking updates from 5.8.0 to 5.10.0
ANY   : Backup stored in '/tmp/onescape/backups/2019-12-17_18:11:19_29405'
INFO  : Updating from 5.8.0 to 5.10.0
INFO  : Incremental update from 5.8.0 to 5.10.0
ANY   : Configuration updated to 5.10.0
```

Successful upgrade saves the target version as a new current configuration version.

Debug Output

The tool provides more detailed information even about individual changes done in the configuration files and status of their application, if run with the debug logging enabled via parameter `--debug`. On the example below, see the **Patch Report** section which uses the format introduced for *diff subcommand* prefixed by patch application status in square brackets:

```
$ onecfg upgrade --debug
DEBUG : Loading migrators
INFO  : Checking updates from 5.4.0 to 5.10.0
DEBUG : Backing up multiple dirs into '/tmp/onescape/backups/2019-12-16_13:16:16_22128'
DEBUG : Backing up /tmp/ONE540/etc/one in /tmp/onescape/backups/2019-12-16_13:16:16_22128/etc/one
```

```
DEBUG : Backing up /tmp/ONE540/var/lib/one/remotes in /tmp/onescape/backups/2019-12-16_13:16:16_22128
DEBUG : Loading migrators
ANY   : Backup stored in '/tmp/onescape/backups/2019-12-16_13:16:16_22128'
DEBUG : Restoring multiple dirs from '/tmp/ONE540'
DEBUG : Restoring /tmp/ONE540/etc/one to /tmp/d20191216-22128-qqek6g/etc/one
DEBUG : Restoring /tmp/ONE540/var/lib/one/remotes to /tmp/d20191216-22128-qqek6g/var/lib/one/remotes
INFO  : Updating from 5.4.0 to 5.10.0
INFO  : Incremental update from 5.4.0 to 5.4.1
DEBUG : 5.4.0 -> 5.4.1 - No Ruby pre_up available
INFO  : Update file '/etc/one/az_driver.conf'
DEBUG : --- PATCH REPORT '/etc/one/az_driver.conf' ---
DEBUG : Patch [OK] set instance_types/ExtraSmall/memory = 0.768
DEBUG : Patch [OK] ins instance_types/Standard_A1_v2 = "cpu"=>1, "memory"=>2.0
DEBUG : Patch [OK] ins instance_types/Standard_A2_v2 = "cpu"=>2, "memory"=>4.0
DEBUG : Patch [OK] ins instance_types/Standard_A4_v2 = "cpu"=>4, "memory"=>8.0
DEBUG : Patch [--] ins instance_types/Standard_A8_v2 = "cpu"=>8, "memory"=>16.0
DEBUG : Patch [--] ins instance_types/Standard_A2m_v2 = "cpu"=>2, "memory"=>16.0
DEBUG : Patch [--] ins instance_types/Standard_A4m_v2 = "cpu"=>4, "memory"=>32.0
DEBUG : Patch [--] ins instance_types/Standard_A8m_v2 = "cpu"=>8, "memory"=>64.0
DEBUG : Patch [--] ins instance_types/Standard_G1 = "cpu"=>2, "memory"=>28.0
...
```

3.3 Troubleshooting

The configuration files upgrade is a complex process, during which many problems may arise. The root cause of all problems is the users' customizations done in the configuration files on places that change in a newer version. Because the upgrade process tries to apply changes from newer versions to existing files, the tool can be confused when it reaches the incompatibly modified part.

In case of a problem, the upgrade process terminates and leaves the state of configuration files unchanged. There is no automatic mechanism preconfigured, but the user has to instruct the tool on how to resolve the problem. This is done by specifying a **patch modes** globally for the whole process, for a particular file, or for a particular file and (intermediate) version we upgrade to.

3.3.1 Patch Modes

The way how the upgrade process works is a typical diff/patch approach. Each version change is described as a series of patches that must be applied. During the patching, some of the following problems may arise:

- A parameter has been removed by the user, but the patch tries to change it.
- Data structure of the parameter isn't the expected one.
- Precise location for change can't be found.

To deal with these situations, there are following patch modes available:

Patch Modes	Action	Problem Cause
`skip`	**Skip** patch operation	Removed or incompatible configuration part.
`force`	Place value in some suitable place, instead of precise place.	No precise place for application found
`replace`	**Replace** user changed values **with new default ones**.	User changed value for which new default appeared

The patch modes are specified using `--patch-modes MODES` parameter passed to the `onecfg upgrade`. Patch modes can be used multiple times, but always the most specific one overrides the more general ones (patch mode for particular file/version overrides the default patch mode). The syntax of the parameter should follow one of the following syntaxes:

- `MODES` - **default patch modes** `MODES` for all files and all versions.
- `MODES:FILE_NAME` - patch modes `MODES` for specific file `FILE_NAME` and all its versions
- `MODES:FILE_NAME:VERSION` - patch modes `MODES` for specific file `FILE_NAME` when upgraded **to version** `VERSION`

Modes (`MODES`) is a comma (`,`) separated list of selected patch modes (**skip**, **force**, **replace**).

Examples

Set default patch mode to **skip** problematic places for all files in any version:

```
# onecfg upgrade --patch-modes skip
```

Set patch mode to **skip** problematic places only for `/etc/one/oned.conf`, leave unspecified mode for all the rest files:

```
# onecfg upgrade --patch-modes skip:/etc/one/oned.conf
```

Set patch mode to **skip** only for `/etc/one/oned.conf` when upgraded **to version** 5.6.0, rest files have unspecified mode:

```
# onecfg upgrade --patch-modes skip:/etc/one/oned.conf:5.6.0
```

Example of multiple patch modes for multiple files:

```
# onecfg upgrade      --patch-modes skip:/etc/one/oned.conf      --patch-modes skip,replace:/etc/one/
```

3.3.2 Restore from Backup

Upgrade operations are done safely on a copy of production configuration files without changing the system state. After upgrade ends successffully, the modified files are copied back to production locations.

Important: Each upgrade operation creates a backup of current directories with OpenNebula configuration files into `/var/lib/one/backups/config/`. In case of error when copying the modified state back to production locations, the automatic restore is triggered.

In the case of a catastrophic failure when even automatic restore fails, the original content of configuration directories must be restored **manually** from initial backup. Example of failed upgrade which requires manual intervention:

```
# onecfg upgrade
ANY   : Backup stored in '/tmp/onescape/backups/2019-12-18_12:22:28_2891'
FATAL : Fatal error on restore, we are very sorry! You have to restore following directories manually
    - copy /tmp/onescape/backups/2019-12-18_12:22:28_2891/etc/one into /etc/one
    - copy /tmp/onescape/backups/2019-12-18_12:22:28_2891/var/lib/one/remotes into /var/lib/one/remot
FATAL : FAILED - Data synchronization failed
```

3.4 OpenNebula Upgrade Workflow

This section describes the typical OpenNebula upgrade process incorporating the OneScape.

Important: For each OpenNebula upgrade (even between minor versions, e.g. 5.10.2 and 5.10.3), configuration files must be processed via 'onecfg upgrade'! If you skip configuration upgrade step for some OpenNebula upgrade, the tool will lose the current version state and you'll have to handle files upgrade manually and *reinitialize* the configuration version management state.

```
# onecfg upgrade
FATAL : FAILED - Configuration can't be processed as it looks outdated!
You must have missed to run 'onecfg update' after previous OpenNebula upgrade.

# onecfg status
--- Versions -------------------------------
OpenNebula:  5.10.1
Config:      5.8.0
ERROR: Configurations metadata are outdated.
```

3.4.1 Step 1 - Get OneScape

Follow the *installation* section to install or **update** the existing OneScape installation.

Note: New OpenNebula major releases require the latest OneScape. You may experience the following error if your OneScape version is too old:

```
# onecfg status
--- Versions -------------------------------
OpenNebula:  5.10.2
Config:      5.8.0
ERROR: Unsupported OpenNebula version 5.10.2
```

3.4.2 Step 2 - Upgrade OpenNebula

Update your OpenNebula packages by following **Upgrading from OpenNebula X.Y** document from official OpenNebula Documentation for the version you are upgrading to.

Important: It's necessary to upgrade your current OpenNebula directly to **5.10.2** or later, which supports the automatic configuration backups.

3.4.3 Step 3 - Update Configurations

This action is usually mentioned in the **Upgrading from OpenNebula X.Y** as a manual step before upgrading the database. OneScape configuration module completely automates the step by running `onecfg upgrade`. Follow *onecfg upgrade* documentation on how to upgrade and troubleshoot the configurations.

Important: Configuration upgrade must be done after each OpenNebula upgrade!

3.4.4 Step 4 - Finish Upgrade

Follow the rest steps from **Upgrading from OpenNebula X.Y** document. It might be necessary to upgrade database, or do some other OpenNebula version-specific steps.

3.4.5 Step 5 - Validation

When all steps are done, run the OpenNebula and check the working state.

Check the configuration state via `onecfg status`. There shouldn't be any errors and no new updates available. Your configuration should be current to the installed OpenNebula version. For example:

```
# onecfg status
--- Versions -----------------------------
OpenNebula:  5.10.2
Config:      5.10.0

--- Available Configuration Updates -------
No updates available.
```

3.5 Appendix - List of Configuration Files

Following table describes all configuration files and their type from directories

- `/etc/one/`
- `/var/lib/one/remotes/`

managed by the **onecfg** tool:

Name	Type
`/etc/one/auth/ldap_auth.conf`	YAML w/ ordered arrays
`/etc/one/auth/server_x509_auth.conf`	YAML
`/etc/one/auth/x509_auth.conf`	YAML
`/etc/one/az_driver.conf`	YAML
`/etc/one/az_driver.default`	Plain file (or XML)
`/etc/one/cli/*.yaml`	YAML w/ ordered arrays
`/etc/one/defaultrc`	Shell
`/etc/one/ec2_driver.conf`	YAML
`/etc/one/ec2_driver.default`	Plain file (or XML)
`/etc/one/ec2query_templates/*.erb`	Plain file (or XML)
`/etc/one/econe.conf`	YAML
`/etc/one/hm/hmrc`	Shell
`/etc/one/oned.conf`	oned.conf-like
`/etc/one/oneflow-server.conf`	YAML
`/etc/one/onegate-server.conf`	YAML
`/etc/one/onehem-server.conf`	YAML
`/etc/one/packet_driver.default`	Plain file (or XML)
`/etc/one/sched.conf`	oned.conf-like
`/etc/one/sunstone-logos.yaml`	YAML w/ ordered arrays
`/etc/one/sunstone-server.conf`	YAML
`/etc/one/sunstone-views.yaml`	YAML

Continued on next page

Table 3.1 – continued from previous page

Name	Type
/etc/one/sunstone-views/**/*.yaml	YAML
/etc/one/tmrc	Shell
/etc/one/vcenter_driver.conf	YAML
/etc/one/vcenter_driver.default	Plain file (or XML)
/etc/one/vmm_exec/vmm_exec_kvm.conf	oned.conf-like
/etc/one/vmm_exec/vmm_exec_vcenter.conf	oned.conf-like
/etc/one/vmm_exec/vmm_execrc	Shell
/var/lib/one/remotes/datastore/ceph/ceph.conf	Shell
/var/lib/one/remotes/etc/datastore/ceph/ceph.conf	Shell
/var/lib/one/remotes/etc/datastore/fs/fs.conf	Shell
/var/lib/one/remotes/etc/im/kvm-probes.d/pci.conf	YAML
/var/lib/one/remotes/etc/im/lxd-probes.d/pci.conf	YAML
/var/lib/one/remotes/etc/market/http/http.conf	Shell
/var/lib/one/remotes/etc/tm/fs_lvm/fs_lvm.conf	Shell
/var/lib/one/remotes/etc/vmm/kvm/kvmrc	Shell
/var/lib/one/remotes/etc/vmm/lxd/lxdrc	YAML
/var/lib/one/remotes/etc/vmm/vcenter/vcenterrc	YAML
/var/lib/one/remotes/etc/vnm/OpenNebulaNetwork.conf	YAML
/var/lib/one/remotes/vmm/kvm/kvmrc	Shell
/var/lib/one/remotes/vnm/OpenNebulaNetwork.conf	YAML

CHAPTER

FOUR

SUPPORT MODULE

4.1 Overview

The aim of the **Support Module** is to provide tool to generate diagnostic bundles with all details necessary handle support cases. It has following functionality:

- Gather configuration state of operating system and services on front-end and nodes.
- Inspect OpenNebula configuration, state and metadata of VMs, hosts, vnets, etc.
- Dump OpenNebula database.
- Scan vCenter user permissions.

Support bundle is easily generated by running command-line tool `onesupport`.

4.1.1 How Should I Read This Chapter

In this chapter, you will find all the information about how to generate support bundles, and what options you can use to limit the content provided into the bundle.

4.2 Basic Usage

The support module comes with 2 specialized tools to

- *generate support diagnostic bundle*,
- *scan vCenter permissions*.

4.2.1 Generate Support Bundle

When there are no special requirements, the support diagnostic bundle can be generated just by running `onesupport` **on the front-end** without any extra parameters. The command must be running under privileged user (directly by root or via sudo). For example:

```
$ sudo onesupport
```

The command generates a diagnostic bundle file and puts it into `/tmp/`. The precise file location is shown at the end of the terminal output, e.g.:

```
Diagnostic archive: /tmp/onesupport.HNpsTCQSb.tar.xz
```

Please **attach this file to the support ticket**.

Important: For most support issues it's enough to provide just a limited support bundle with front-end only specific information and without database dumps. The limited bundle can be generated by running:

```
$ sudo onesupport frontend nodb
```

The tool prints all actions, based on which you can decide if you want to provide such a bundle to the support team. Example full output:

```
$ sudo onesupport
    ___  _ __   ___
   / _ \| '_ \ / _ \
  | (_) | | | |  __/     OpenNebula Support Tool
   \___/|_| |_|\___|

 ------------------------------------------------------
 Group: all
 - with conf: true
 - with logs: true
 - with DB:   true

Get OS distribution
Get current user
Get user "oneadmin"
Get installed packages
Get kernel version
Get security settings
Get memory/swap
Get Ruby version and Gems
Get CPU information
Get system services
Get process list
Get kernel runtime parameters (sysctl)
Get mounts
Detect web server
Get OpenNebula version
Get OpenNebula running processes
Detect changes in OpenNebula packages
Get OpenNebula database information
Get OpenNebula hosts
Get OpenNebula logs
Get system logs
Dump OpenNebula database
Get OpenNebula configuration
Get OpenNebula remotes
Dump OpenNebula objects
  - hosts
  - vnets
  - datastores
  - clusters
  - images
  - templates
  - ACLs
  - VDCs
  - OneFlow templates
Dump OpenNebula instance objects
```

```
  - VMs
  - OneFlow instances
Get web server configuration
Inspecting the OpenNebula hosts
  - KVM host localhost (oneadmin with oneadmin's key)
[localhost] Get OS distribution
[localhost] Get current user
[localhost] Get user "oneadmin"
[localhost] Get installed packages
[localhost] Get kernel version
[localhost] Get security settings
[localhost] Get memory/swap
[localhost] Get Ruby version and Gems
[localhost] Get CPU information
[localhost] Get system services
[localhost] Get process list
[localhost] Get kernel runtime parameters (sysctl)
[localhost] Get mounts
[localhost] Get IP/bridge runtime config.
[localhost] Get firewall config
[localhost] Get Open vSwitch runtime config.
[localhost] Detect libvirt
[localhost] Get libvirt domains list
[localhost] Get libvirt domains XML and screenshots
[localhost] Get libvirt configuration
[localhost] Get system logs

Diagnostic archive: /tmp/onesupport.HNpsTCQSb.tar.xz
```

Follow the *advanced usage* guide to learn how to limit the information provided within the support bundle or how to use it on different hosts then frontend.

4.2.2 Scan vCenter Permissions

When you are using OpenNebula cloud to manage vCenter infrastructure, it might be necessary to know also details about the permissions configuration inside vCenter. This is **automatically got during the support bundle preparation**, but can be scanned separately at any time (e.g., in case the automatic run fails).

Dedicated tool `onesupport_vcenter_privs` is used to gather permissions configuration.

Important: The tool connects directly to your vCenter instance and must be provided with connection parameters.

Syntax:

```
$ onesupport_vcenter_privs
Usage: onesupport_vcenter_privs [arguments]

Mandatory arguments:
  --host=name       .... vCenter hostname
  --user=name       .... vCenter login user name
  --password=text   .... vCenter password
  --check-user=name .... vCenter user for OpenNebula to check
```

Example run:

```
$ onesupport_vcenter_privs --host=vcenter.localdomain      --user=administrator@vsphere.local --passw
```

The result is printed on the terminal (standard output). Can be redirected to the file, or just copy&pasted into the support ticket.

4.3 Advanced Usage

This section describes all options on how to adjust the generation of the support bundle.

4.3.1 Generate Support Bundle

The purpose of the `onesupport` tool is to gather as much as possible information about the environment so that customer support can give more accurate and faster responses. The usage is very simple, there are only a few configuration options. All are described in brief documentation available via argument `--help`.

```
$ sudo onesupport --help
onesupport [host types] [dump types]

Host types:
  all           ... start on frontend and inspect all hosts (default)
  frontend      ... gather only frontend specific data
  host          ... gather only KVM host specific data

Dump types:
  nodb, db      ... (don't) dump database (ONE)
  noconf, conf  ... (don't) dump configuration (ONE, libvirt, Apache/NGINX)
  nologs, logs  ... (don't) dump logs (ONE and system logs)
```

There are 2 types of arguments to specify:

- **host type**
- **dump type**

Host Types

What data are gathered depends mainly on the type of host we are running tool on. Each host type gets same common data (operating system, hardware, memory, installed software packages, system services, mounts, logs etc.) and differs only in data specific for the type.

Available options are:

Option	Main Subject
`frontend`	OpenNebula front-end services configuration and state, database dump, various entities (e.g., VMs).
`host`	Virtualization services (libvirt, KVM) and network configuration.
`all`	Combination of `frontend` mode and `host` mode. It starts with front-end specific data and connects to each virtualization host to get host-specific data. This mode is the **default**.

Examples

Simple run gathers all information (runs are equivalent):

```
$ sudo onesupport
$ sudo onesupport all
```

Get only front-end specific data, must be run on front-end:

```
$ sudo onesupport frontend
```

Get only host-specific data, must be run on virtualization host:

```
$ sudo onesupport host
```

Dump Types

Level of detail contained in the gathered data can be adjusted by dump type parameters. Following dump types are supported:

Option	Description
`db`, `nodb`	Enable / disable database dumps.
`conf`, `noconf`	Enable / disable bundling of configuration files.
`logs`, `nologs`	Enable / disable bundling of logs.

All dump types are enabled by default (`db conf logs`), but can be selectively disabled with negative options `nodb`, `noconf` and/or `nologs`.

Important: If positive dump types (`db`, `conf`, `logs`) are used on the command line, only the specified types are gathered and no other ones.

If negative dump types (`nodb`, `noconf`, `nologs`) are used, these types are excluded from the support bundle. All the rest types are included.

Examples

Simple run gathers all information (runs are equivalent):

```
$ sudo onesupport
$ sudo onesupport db conf logs
```

Get support bundle without any database dumps and logs:

```
$ sudo onesupport nodb nologs
```

Get support bundle with database dump, but no logs and configurations:

```
$ sudo onesupport db
```

Dump types and host types parameters can be combined

```
$ sudo onesupport frontend nodb
```