

Open Source MANO

by ETSI

OSM White Paper

OSM RELEASE FOURTEEN RELEASE NOTES

July 2023

Open Source MANO

Technical Steering Committee

ETSI
06921 Sophia Antipolis CEDEX, France
Tel +33 4 92 94 42 00
info@etsi.org
www.etsi.org

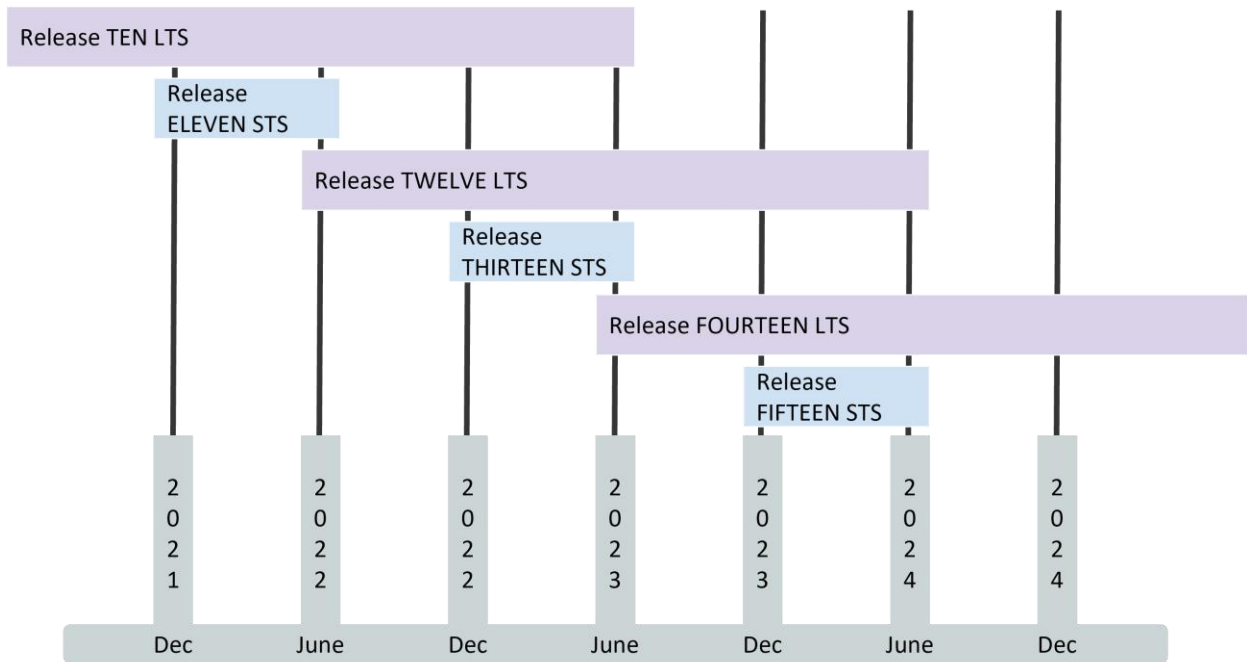


Contents

Introduction	3
Closed-loop life cycle architecture	4
Adoption of new monitoring architecture for closed loops	4
Service KPI of VNF using Prometheus exporter endpoints	6
Auto-heal and auto-scale switch	7
Security and stability enhancements	8
Replacement of Pycrypto library with PycryptoDome	8
Enforcement of standard pod security policies to Helm-based EE namespaces	8
Authenticated gRPC for Helm-based Execution Environments	8
Updated dependencies to meet LTS needs	9
Enhancements in OSM code thanks to Coverity analysis tool	9
Usability and platform management	10
Audit Logs Generation	10
User Management Enhancements	11
Infra modelling and NF lifecycle	12
RO performance optimizations for the polling of VM status	12
Enhanced IPv6 network creation	12
Static IPv6 (Dual-Stack) Assignment for VNFs	12
Transport API (T-API) WIM connector	13
Support of volume multi-attach	13
Use of existing flavor identifier as instantiation parameter	13
Instantiation parameters for Juju bundles	13
OSM installation	14
Helm Charts for deploying OSM on K8s	14
Upgrade of OSM services with helm	14
OSM client enhancements	15

Introduction

The ETSI Open Source MANO community is proud to announce OSM Release FOURTEEN, meeting our already established cadence of two releases per year, alternating between LTS releases (2 years support) and Standard releases (6 months support).



OSM release cadence

Release FOURTEEN is an LTS release of ETSI OSM, providing two years of continuous support with bug fixes and security patches and including significant improvements in many key areas.

Closed-loop life cycle architecture

- GA of new monitoring architecture for closed loops.
- Service KPI of VNF using exporter endpoint.
- Autoheal switch and autoscale switch.



Infra modelling and NF lifecycle

- RO performance optimization.
- Simultaneous IPv4 and IPv6 support.
- Transport API (TAPI) WIM connector
- Support of volume multi-attach.
- Use existing flavor-id as an instantiation parameter.
- Instantiation parameters for Juju bundles.



Security enhancements

- Replacement Pycrypto with PycryptoDome.
- Pod Admission Policy for Helm-based EE.
- Authenticated gRPC for Helm-based EE.



OSM installation

- Helm Charts for deploying OSM on K8s.
- Update/Upgrade of OSM services.



Usability and platform management

- User management enhancements.
- Audit logs generation for OSM



OSM client

- Support of different output formats.
- Replacement of Pycurl with Requests



Release FOURTEEN - Feature summary

Closed-loop life cycle architecture

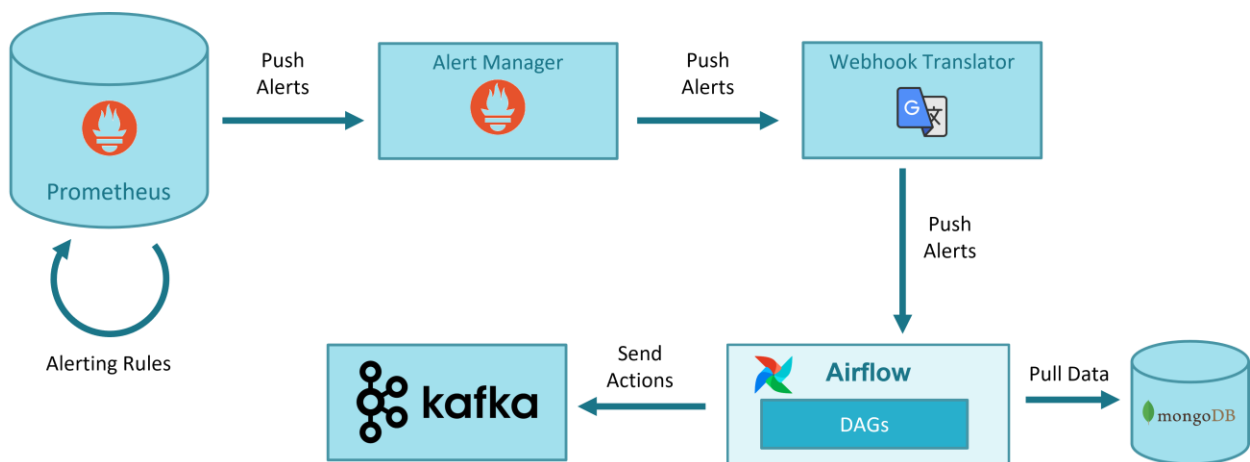
Adoption of new monitoring architecture for closed loops

OSM Release THIRTEEN introduced a new scalable architecture for Service Assurance (SA) based on Apache Airflow and Prometheus Stack. It incorporated new workflows for getting the state of Network Functions (NF), Network Services (NS) and VIM, and set the foundations for building closed loops.

With OSM Release FOURTEEN, MON and POL functionality has been transferred to the new SA architecture, which is enabled by default with OSM installation. Specifically, Release FOURTEEN includes the following developments:

- Metric acquisition (VM resource consumption)
- Closed-loop for auto-healing
- Closed-loop for auto-scaling
- VNF alarms

The following figure illustrates the general workflow for closed-loops in OSM that is used both for auto-healing and auto-scaling.



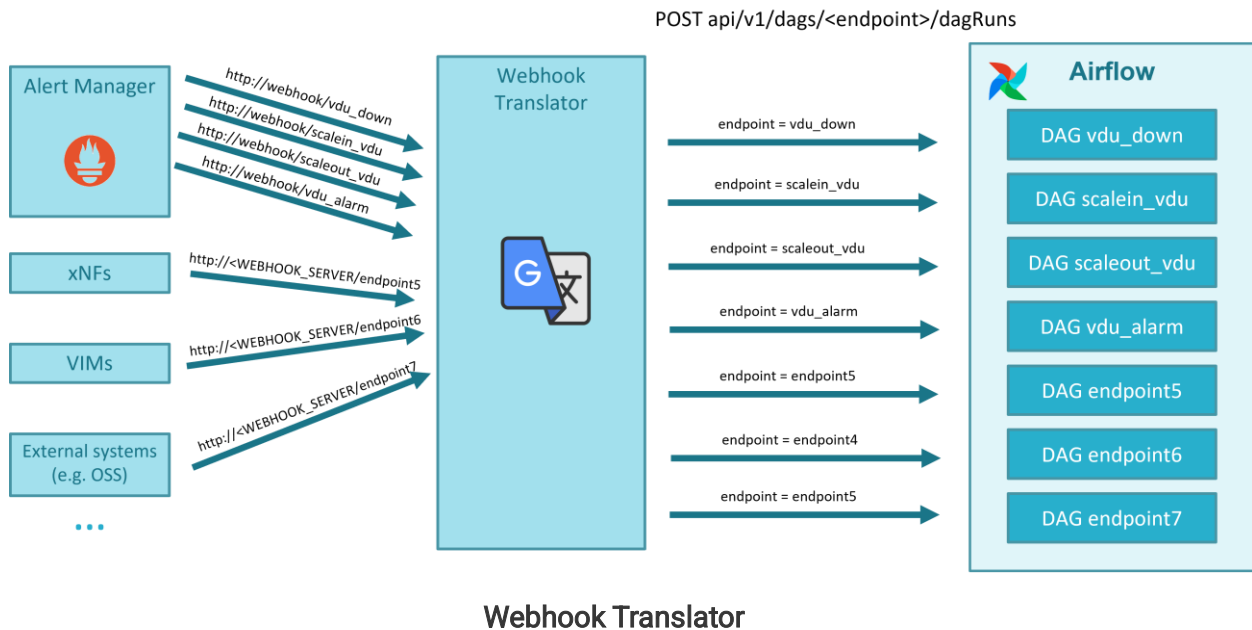
Generic workflow for closed loops in OSM

- **Prometheus alerts.** All monitoring information (status of VIMs and VNFs managed by OSM) is stored in Prometheus. Prometheus has a simple alert generation mechanism based on [alerting rules](#) that are configured in a similar way as the recording rules. The alerts are automatically triggered and stopped depending on their defining rule.
 - For the case of auto-healing operation, a static rule is pre-defined, which generates an alert when the status of a VM is down for more than a given period.
 - For the case of auto-scaling operations, a sidecar container deployed with Prometheus dynamically updates the Prometheus alert rules as they are created/deleted by OSM in MongoDB at instantiation/termination time.

- **AlertManager.** [AlertManager](#) allows managing alerts, including silencing, inhibition, aggregation and sending out notifications. The alerts generated by Prometheus are sent to AlertManager, which forwards them to a webhook. The use of AlertManager is necessary as an intermediate step because Prometheus cannot send alerts directly to a webhook. In addition, AlertManager includes mechanisms for silencing, inhibition, aggregation, etc., which could be leveraged by OSM in the future.
- **Webhook Translator.** AlertManager can send alerts to multiple external systems via, for example, HTTP, but does not integrate directly with Airflow, which is the system that will run the workflows or DAGs for healing and scaling. Airflow has its own format for webhooks. The Webhook Translator is added to translate HTTP requests: it receives an HTTP request from AlertManager and forwards it to a supported Airflow webhook in the format expected by Airflow.
- **Airflow DAG driven by webhook.** The last step in the pipeline is running the DAG which will process the alert and execute the necessary actions to carry out the closed-loop operation. In the case of auto-healing, when the firing alert about a VM arrives, the DAG will check if there is an auto-healing rule stored in MongoDB regarding to this VM/VDU. In that case, the DAG updates alert status in MongoDB and sends to Kafka a heal message that will be consumed by LCM, which finally performs the healing operation.
- **MongoDB.** The information (auto-healing and auto-scaling rules) used for closed-loop operations is stored in MongoDB in the collection alerts. LCM is responsible of inserting and deleting the alarm-related objects when an NS is instantiated or terminated.

The Webhook Translator is a new component that has been added to translate webhooks, allowing the translation of an alert generated by Alert Manager to the appropriate Airflow DAG endpoint, and enabling the capability to translate in the future external alerts from VNF or other systems to dedicated Airflow workflows. The main characteristics of the Webhook Translator are the following:

- It is lightweight. A very small number of lines of code does the work.
- It is stateless. It only translates HTTP requests. No state for those translations. When running as a deployment, native scaling is achieved by means of Kubernetes services.
- It is simple. It is based on [FastAPI](#), a simple and fast framework for developing an HTTP REST API in Python.
- It is independent from the source of the alert.
- No maintenance is expected.



Service KPI of VNF using Prometheus exporter endpoints

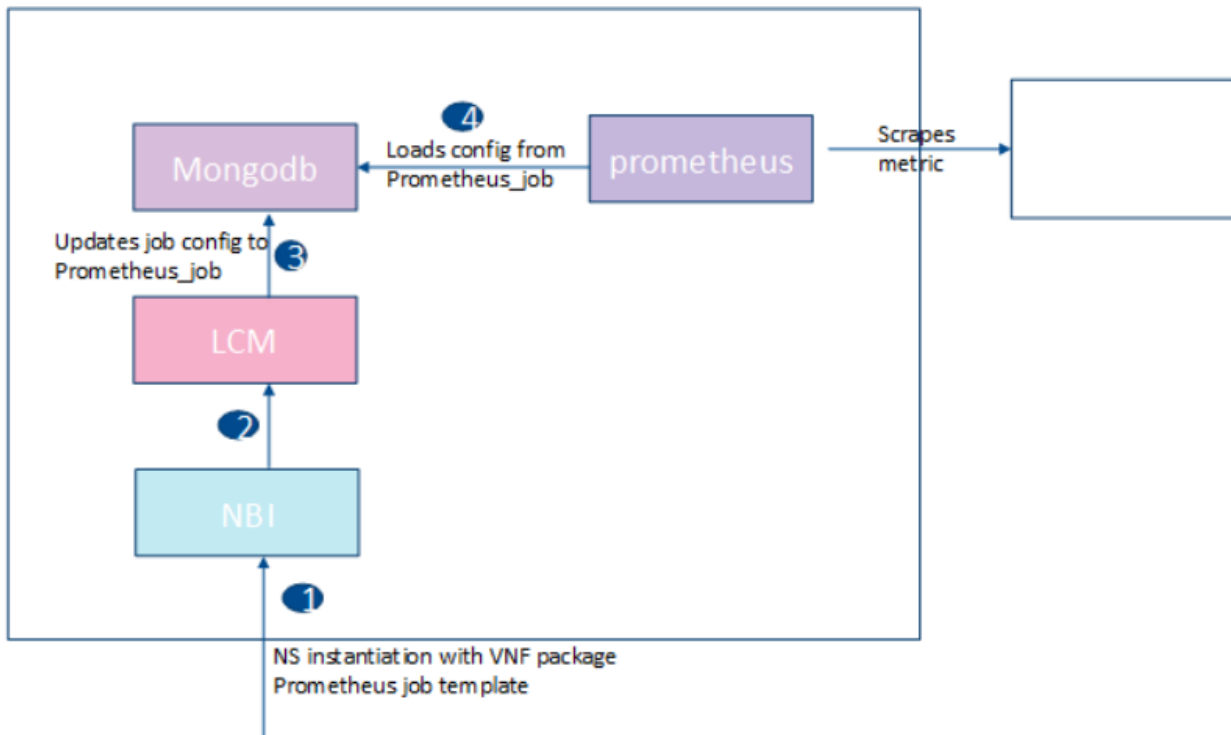
In previous releases of OSM, SNMP metrics from a Network Function (NF) could be retrieved using an SNMP exporter bundled in the NF package. Release FOURTEEN generalizes the previous concept and enables the gathering of any NF metrics, also called service KPI metrics, as long as they are exposed with a Prometheus exporter.

This feature adds the exporter-endpoint option to the NF descriptor, where users can provide the endpoint of the node exporter, thus allowing OSM (Prometheus component) to scrape metrics directly from the NF.

Below an example of the relevant excerpt of a NF descriptor where the exporter-endpoint is specified:

```
vnfd:
df:
  exporters-endpoints:
    metric-path: /metrics
    metric-port: 9100
    external-connection-point-ref: vnf-cp0-ext
```

NF packages are bundled with a Prometheus job template, and this template is stored in MongoDB. At instantiation time, OSM will update the scraping jobs in Prometheus with the new job from the template, thus allowing Prometheus to collect the service KPI metrics of the NF.



Workflow to update Prometheus jobs in OSM

By monitoring and analyzing these service KPI metrics, network operators can gain insight into the performance of NFs and take proactive steps to optimize service delivery and ensure that quality of service meets the expectations.

This feature will improve the integration of NF metrics for its use in OSM, giving the wide adoption of Prometheus framework for the monitoring of CNF. Specifically, most of the CNF come with node exporters exposing NF metrics, which now can be easily modeled and incorporated into OSM.

Auto-heal and auto-scale switch

With the new Service Assurance architecture, Airflow DAGs for scaling can be selectively disabled in Airflow UI by pressing the toggle next to the DAG to pause/unpause it:

- scalein_vdu, to enable/disable auto-scale-in
- scaleout_vdu, to enable/disable auto-scale-out

In setups where the old Service Assurance based on MON and POL is used, it is possible to enable/disable auto-scaling and auto-healing by patching the POL deployment in kubernetes, setting the environment variables OSMPOL_AUTOSCALE_ENABLED and OSMPOL_AUTOHEAL_ENABLED to True or False

Disabling auto-healing and auto-scaling might be highly convenient in production environments in circumstances such as the following:

- During the upgrade/maintenance of compute nodes in a datacenter, where the compute nodes may go down and VNFs running in those compute nodes will move to an “ERROR” state.
- During NF upgrade operations, particularly in NF with huge databases such as HSS, NPDB, etc., that require multiple restarts with a high chance that the NF may stuck in “ERROR” state.
- During NF upgrade, where the CPU utilization of the VDUs may reach peak values leading to an undesired scale-out.
- During resource crunch scenarios, i.e. when there are no resources in the DC to scale out the number of VDUs of a NF.

Security and stability enhancements

Replacement of Pycrypto library with Pycryptodome

OSM was using unmaintained Pycrypto module which is vulnerable to heap-based buffer overflow and allows remote attackers to execute arbitrary code in the Python application. This feature fixed the vulnerability by replacing PyCrypto module with PyCryptodome. Within this feature, the desired mode of Symmetric ciphers has to explicitly provided the reason why ECB is not the default mode anymore. Pycryptodome supports the traditional modes of symmetric ciphers such as ECB, CBC, CFB, OFB, CTR, OpenPGP (a variant of CFB, RFC4880). Additionally, it supports the authenticated encryption methods such as CCM (AES only), EAX, GCM (AES only), SIV (AES only), OCB (AES only), ChaCha20-Poly1305. Besides, asynchronous encrypt/decrypt methods are added to common module which are ready to be used by other modules.

Enforcement of standard pod security policies to Helm-based EE namespaces

When using Helm based Execution Environments (EE) for configuring a Network Function, the provided helm chart could create pods without restricting any capability. For instance, pods can request hostpaths, privilege escalation, etc.

This feature uses the built-in Pod Security Admission controller of Kubernetes to enforce a baseline security policy and prevent the creation of pods with risky capabilities. To override the baseline Standard Policy, the environment variable `OSMLCM_VCA_EEGRPC_POD_ADMISSION_POLICY` can be set in the LCM pod to privileged or restricted according to the requirements.

Authenticated gRPC for Helm-based Execution Environments

In previous releases, the communication between LCM and Helm-based Execution Environments (EE) was based on plain-text gRPC. This feature adds mutual (client and server) TLS to gRPC channels.

TLS certificates are handled by cert-manager and a self-signed Certificate Authority deployed when OSM is installed. The Certificate Authority is used to sign certificates for every Network Service instance, thus guaranteeing that only LCM can access gRPC servers in the EE.

Backwards compatibility is guaranteed by falling back to plain-text gRPC if the TLS communication fails. TLS can be strictly enforced by setting the environment variable `OSMLCM_VCA_EEGRPC_TLS_ENFORCE` to True in LCM pod.

Updated dependencies to meet LTS needs

With the availability of OSM FOURTEEN LTS as a two year commitment, the base software for OSM containers has been updated as follows:

- Python 3.10
- Ubuntu 22.04
- Kubernetes 1.26
- Helm 3.11.3
- Juju 2.9.42

Enhancements in OSM code thanks to Coverity analysis tool

Coverity is a static analysis tool that helps you find and fix software defects. It can find a wide range of defects, including security vulnerabilities, memory leaks, and coding errors, and can help to comply with industry security standards.

Below some of the fixes discovered with Coverity that have been fixed in OSM release FOURTEEN:

- Coverity-CWE 330:
 - Summary: use of insufficiently random values.
 - Details: the `randint()` function has been replaced with `random.SystemRandom().randint()` to improve security.
 - Coverity-CWE 22:
 - Summary: improper limitation of a pathname to a restricted directory ('path traversal').
 - Details: the path traversal vulnerability has been fixed by limiting the path to a restricted directory.
 - Coverity-CWE 260:
 - Summary: password in configuration files.
 - Details: the hardcoded credentials in the `cloud-config.txt` file have been removed as they are not being used anywhere in the test case.
 - Coverity-CWE 398:
 - Summary: 7PK - Code Quality.
 - Details: the copy-paste error has been fixed by removing the repetitive if condition and replacing it with a single if condition.
 - Coverity-CWE 476:
 - Summary: NULL pointer dereference.
-

- Details: the NULL pointer dereference vulnerability has been fixed by checking for the null value before dereferencing it.
- Coverity-CWE 295:
 - Summary: improper certificate validation.
 - Details: the SSL certificate validation has been enabled.
- Coverity-CWE 569:
 - Summary: expression issues.
 - Details: the if condition has been fixed to check for the same value on both sides.
- Coverity-CWE 561:
 - Summary: dead code.
 - The logically dead code has been removed.

Usability and platform management

Audit Logs Generation

OSM logs audit events to record actions that answer the question of **“Who did what, when, and where?”**. Audit logs record the occurrence of an event, the operation performed by the event, the time at which the event occurred, and the user/project that performed the event in a system.

Audit logs are recorded in the NBI module and they follow Common Event Format (CEF). CEF is a standardized logging format to structure logs in a common format that could simplify logging and enable the integration of logs in to a single management system.

The following audit logs are available:

1. Incorrect login attempt- Records any user incorrect login attempts to OSM.
2. User Login and Logout- Records any user login and logout operations in OSM.
3. Resetting Passwords- Records changes in user account password
4. Administrator access- Records any access attempts to accounts that have system privileges.
5. Account administration/Services- Records all account activity like fetching, creating, updating, or deleting resources from OSM

All the recorded audit logs follow the format below:

```
CEF:Version|Device Vendor|Device Product|Device Version|Name|Severity|Extension
```

A sample CEF log for User login would be as below:

```
CEF:0|OSM|OSM|14.0.0|User Login|1|msg=User Logged In, Project\=admin Outcome\=Success suser=admin
```

User Management Enhancements

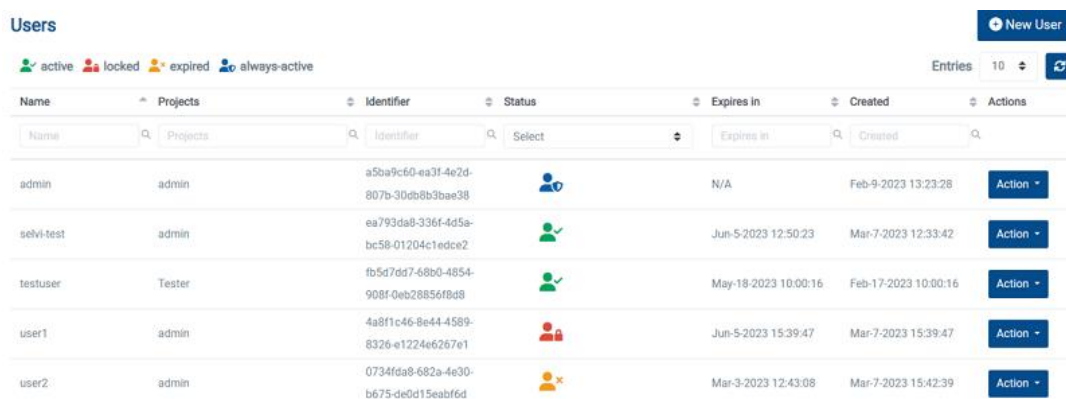
This feature provides additional security enhancements for user management in the NBI Module, as follows:

- Enforcing password change on first login of an user
- Account expiration policies to allow administrators managing user accounts efficiently.
- Password notifications reminders for end users to update their passwords regularly, making it harder for attackers to crack weak or old passwords.
- Centralized enforcement of user management policies in the NBI module, making it easier for administrators to manage large numbers of users.

By default, user management enhancements to enforce those best practices are enabled in OSM. The default behaviour includes the following:

- Password expiration after 30 days, then forcing the user to renew the password.
- User account expiration 90 days after the user is created, then forcing a system admin to renew the account.
- Limited number of consecutive failed login attempts set to 5. After that, if a user fails to login, the account will be locked.

Password renewal and user account lock/unlock can be managed both from OSM client and from the UI.



Name	Projects	Identifier	Status	Expires in	Created	Actions
admin	admin	a50a9c60-aa3f-4e2d-807b-30db8b3bae38		N/A	Feb-9-2023 13:23:28	Action
selvi-test	admin	ea7993da8-336f-4d5a-bc58-01204c1edce2		Jun-5-2023 12:50:23	Mar-7-2023 12:33:42	Action
testuser	Tester	fb5d7d7-68b0-4854-908f-0eb28856fb48		May-18-2023 10:00:16	Feb-17-2023 10:00:16	Action
user1	admin	4a8f1c46-8e44-4589-8326-e1224e6267e1		Jun-5-2023 15:39:47	Mar-7-2023 15:39:47	Action
user2	admin	0734fda8-682a-4e30-b675-de0d15eabf6d		Mar-3-2023 12:43:08	Mar-7-2023 15:42:39	Action

Enhanced user management in OSM UI

The default behaviour can be globally enabled/disabled by patching the NBI deployment or configmap, setting the environment variable OSMNBI_USER_MANAGEMENT to True or False.

Infra modelling and NF lifecycle

RO performance optimizations for the polling of VM status

RO module updates the VDU details in VNF records periodically unless disabled by configuration. For Openstack VIM, RO was updating the VM status based on `vm_id` which is kept in the `completed_ro_tasks` collection in the MongoDB. For each virtual machine, it was sending several requests to the Openstack Cloud APIs to get VM details, networks and ports details. Withing this feature, the VM status checks are performed using a more efficient mechanism than individual VM query for Openstack VIM Connector. RO executes a monitoring job which is executed periodically according to `refresh_active` period. This job collects all the VM details and port details belongs to a VIM account by sending only two requests to Openstack cloud APIs. Then, collected bulk data is parsed and all the active VM statuses are updated. This improvement in VM polling mechanism for Openstack, prevents hammering Cloud APIs if there are numerous VMs need to be monitored.

Enhanced IPv6 network creation

This feature enhances network creation in OSM by allowing NF and NS designers to include `ipv6-address-mode` as part of the Virtual Link Descriptors, being possible to set it to `slaac`, `dhcpv6-stateful`, or `dhcpv6-stateless`. With this parameter, it is possible to control the way VIMs are assigning IPv6 addresses and advertising them to the VMs. Currently the parameters are only taken into account in Openstack environments, but its extension to other types of VIM will be considered for future releases.

Static IPv6 (Dual-Stack) Assignment for VNFs

This feature enables configuring and assigning static IPv6 addresses to VNFs.

Typically, IP addresses are provided as instantiation parameters in OSM. However, in some circumstances, it could be useful to configure static IPv6 and IPv4 addresses in the NS Descriptor. In previous releases, OSM allowed assigning static IPv4 addresses to VNFs or assigning dynamic IPv4 and IPv6 addresses using DHCP. In this release, static IPv6 address can be configured in the NS Descriptor in addition to the static IPv4 already available, thus making static dual stack assignment possible.

Below an example of the relevant excerpt of a NS descriptor where the dual stack IP addresses are configured under “`ip-address`”:

```
virtual-link-connectivity:  
- constituent-cpd-id:  
  - constituent-base-element-id: vnf  
    constituent-cpd-id: vnf-cp0-ext  
  ip-address:  
    - 192.168.1.20  
    - 2001:db8::23e
```

Static Dual-Stack IP assignment is only supported for VNFs deployed in Openstack VIM.

Transport API (T-API) WIM connector

Transport API (T-API) is a standard API defined by the Open Networking Foundation (ONF) (<https://wiki.opennetworking.org/display/OTCC/TAPI>) to provide WAN infrastructure management capabilities over multi-domain SDN networks.

OSM Release FOURTEEN includes a new WIM Connector to interact with WIMs based on T-API for the creation of inter-DC networks in multi-site network scenarios.

Support of volume multi-attach

This feature allows OSM to share a volume between multiple VDUs of a VNF, by setting the key `multiattach` to `True` in `vduStorageRequirements`. This is required for some NFs in production that rely on the shared-volume mechanism to exchange information between VDUs or to guarantee in some HA scenarios that all VDU replicas have access to the same volume. Currently the parameters are only taken into account in Openstack environments, but its extension to other types of VIM will be considered for future releases.

Use of existing flavor identifier as instantiation parameter

This feature allows the usage of a specific flavor provided at instantiation time. With this feature, manually created flavors with unmodeled properties in OSM Information Model can be used for VM creation.

Since OSM's birth, the Information Model for NF Descriptors has tried to capture in a VIM-agnostic way any potential parameter required for a VDU flavor. In public clouds, where the flavors cannot be created, OSM searches the flavor with less resources that meets the requirements. In private clouds like Openstack-based, OSM creates the flavor.

The best practice is to model the NF descriptor with the existing parameters, leaving OSM the creation/selectin of a flavor meeting the requirements. However, we acknowledge there might be special situations in production where an existing flavor needs to be used. This is the case for some specific NF vendors, which requests some metadata/properties to be part of the flavor. This feature has been included in this release to support those scenarios.

Instantiation parameters for Juju bundles

OSM is capable of deploying KNFs that use Juju bundles as KDUs. However, any desired personalization of the Juju bundle configuration required a modification of the original KNF package. This feature enhances the capabilities of OSM regarding the deployment of KNFs. OSM is now able to add custom parameters to Juju bundles upon NS instantiation without any adjustment on the original set of validated packages.

This feature eases the deployment of personalized solutions based on existing Juju Bundles. It allows you to customize settings in an upstream bundle for your own needs: you can modify the number of units to deploy or set custom machine constraints at instantiation time. However, OSM will not allow you to add new applications. All the applications in the instantiation parameters must exist on the original Juju bundle.

OSM installation

Helm Charts for deploying OSM on K8s

Since Release FOURTEEN, the deployment of OSM services (LCM, RO, NBI, NG-UI, etc.) in the community installer is done with a Helm chart.

When OSM is installed, behind the scenes the following steps are done by:

- Installation of local LXD server (required for LXD-based proxy charms)
- Installation of Docker CE
- Installation and initialization of a local Kubernetes cluster, including a CNI (Flannel), container storage (OpenEBS), Load Balancer (MetalLB) and a certificate manager (cert-manager).
- Installation of Juju client
- Bootstrapping of juju controller to allow the deployment of Execution Environments in local LXD server and local Kubernetes cluster
- Deployment of OSM
 - Deployment of Mongo DB charm with juju
 - Deployment of services with the OSM Helm Chart, which includes the OSM components (NBI, LCM, RO, NG-UI, MON, webhook translator) and other services (Mysql, Keystone, Zookeeper, Kafka, Prometheus, Grafana)
 - Deployment of NG-SA (new Service Assurance), which includes Airflow, Prometheus Alert Manager and Prometheus Pushgateway Helm Charts
- Installation of OSM client

This feature covers specifically the **deployment of services with the OSM Helm Chart**. An OSM helm chart has been created, encompassing the Kubernetes manifests that existed in previous releases and restructuring them conveniently, following cloud-native best practices regarding the use of configmaps and secrets to configure the different components. In addition, the helm chart comes with a values file that allows, among other configuration options, the following:

- Enable/disable modules at will through a boolean flag
- Change the log level globally or per module
- Change the docker repository, image and tag globally or for each module
- Set specific configuration options per module

Upgrade of OSM services with helm

Once a Helm chart has been created to deploy OSM services, that Helm Chart and future versions of it can be used to easily upgrade OSM.

The procedure for upgrading OSM is as simple as:

- Getting the source code of the helm chart from the [OSM devops repository](#)
- Checking out the desired version
- Getting the current values used in the running helm release
- Update values conveniently with the desired options
- Run a `helm upgrade` command using the helm chart from the repo and the updated values file.

OSM client enhancements

OSM Release FOURTEEN includes some improvements in the OSM client, such as the support of different output formats for some commands and the replacement of Pycurl library by the well-known Requests library for HTTP interaction.

Both features were developed by participants of the recent [OSM Hackfest for developers held in Castelldefels in June 2023](#).



ETSI
06921 Sophia Antipolis CEDEX, France
Tel +33 4 92 94 42 00
info@etsi.org
www.etsi.org

This White Paper is issued for information only. It does not constitute an official or agreed position of ETSI, nor of its Members. The views expressed are entirely those of the author(s).

ETSI declines all responsibility for any errors and any loss or damage resulting from use of the contents of this White Paper.

ETSI also declines responsibility for any infringement of any third party's Intellectual Property Rights (IPR), but will be pleased to acknowledge any IPR and correct any infringement of which it is advised.

Copyright Notification

Copying or reproduction in whole is permitted if the copy is complete and unchanged (including this copyright statement).

© European Telecommunications Standards Institute 2019. All rights reserved.

DECT™, PLUGTESTS™, UMTS™, TIPHON™, IMS™, INTEROPOLIS™, FORAPOLIS™, and the TIPHON and ETSI logos are Trade Marks of ETSI registered for the benefit of its Members.

3GPP™ and LTE™ are Trade Marks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.

GSM™, the Global System for Mobile communication, is a registered Trade Mark of the GSM Association.