



Common Management Tools Across Network, Storage and Compute Product/RFI Requirements

Version 1.1

A white paper from the
ONUG Common Management Tools
Across Network, Storage and
Compute Working Group

May, 2015

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



**Open Networking
USER GROUP**

Definition of Open Networking

Open networking is a suite of interoperable software and/or hardware that delivers choice and design options to IT business leaders, service and cloud providers. At its core, open networking is the separation or decoupling of specialized network hardware and software – all in an effort to give IT architects options in the way in which they choose to design, provision, and manage their networks. These technologies must be based on industry standards. The standards can be de-facto as adopted by a large consortium of the vendor community, open in the sense that they are community based, or defined as standards by the prevailing standards bodies. Open networking hopes to deliver on two promises:

- 1) Decoupling of network hardware and software which mitigates vendor lock-in and shifts network architecture structure options to users
- 2) Significant reduction of the total cost of ownership model, especially operational expense

Scope

The following document has been created to identify Open Networking User Group's (ONUG's) use case for Common Management/Monitoring Tools across Network, Compute and Storage systems. The goal is to define requirements, guidelines and overall framework that could not only help the user community to obtain a vehicle for requesting open interoperable solutions from vendors, and open source communities, but also for those communities to understand the requirements to provide answers.

ONUG's goal is to facilitate the exchange between communities, identifying opportunities for creation of new open environments of interoperability and reusability of solutions. The bar is set high for this particular ONUG use case as the working group not only looks at one specific discipline, but at multiple domains, trying to understand requirements across multiple solution elements from the monitoring and management perspective.

Methodology, Scope and Reach

At the outset, this working group feels the need to clarify the intent and scope of this effort. As the name of the use case suggests, the opportunities and reach could be very broad, and further, to avoid becoming too generic or overlap other important initiatives, this working group has established the following basis for the scope of this effort:

1. The working group could have focused the scope on a specific operating system (OS) or a hypervisor or embedded systems, but it wanted to remain OS-agnostic and, instead, concentrated on tools that can be openly used for all of them. In addition, the scope of this working group focuses on the requirements across the network, storage and compute domains equally.
2. The goal is to use technologies and solutions based on open development concepts/licenses to identify tools to monitor, manage and collect metrics, analytics, forensics and other insightful information from physical or virtual devices to help perform event management, troubleshooting and self-remediation across network, compute and storage domains, keeping the key attributes of openness and transparency as the vehicle to realize interoperability.
3. On the management part of the use case title, the group did not want to overlap open orchestration/management efforts of groups such as OpenStack, OpenFlow and OpenDaylight.
4. The concept applies not only to physical bare metal, switches and routers, but encompasses a broad sweep of P+V (Physical + Virtual) appliances, servers and devices, which have many elements in common, although the challenge may be the commonalities on the software side.
5. The scope requires the use of open standards and open tools, including the possible adoption and standardization of northbound application programming interfaces (APIs).

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



**Open Networking
USER GROUP**

Open Networking User Group (ONUG)

ONUG is one of the largest industry user groups in the networking and storage sectors. Its board is made up exclusively of IT business leaders, with representation from Fidelity Investments, FedEx, Bank of America, UBS, Cigna, Pfizer, JPMorgan Chase, Citigroup, Credit Suisse, Gap, and Symantec. The ONUG mission is to guide and accelerate the adoption of open networking solutions that meet user requirements as defined through use cases, proof of concepts, hackathons, and deployment examples to ensure open networking promises are kept.

The ONUG community is led by IT business leaders and aims to drive industry dialogue to set the technology direction and agenda with vendors. To that end, ONUG hosts two major conferences per year where use cases are defined and members vote to establish a prioritized list of early adopter, open networking projects that communicate propensity to buy and budget development. The vendor community stages proof of concepts based upon ONUG Use Cases, while standards and open source organizations prioritize their initiatives and investments based upon them. ONUG also hosts user summits and smaller, regional user-focused Fireside Chat Meet-Ups through the year. ONUG defines six architectural areas that will open the networking industry and deliver choice and design options. To enable an open networking ecosystem, a common multivendor approach is necessary for the following six architecture components::

- 1) Device discovery, provisioning, and asset registration for physical and virtual devices
- 2) Automated “no hands on keyboards” configuration and change management tools that align DevOps and NetOps
- 3) A common controller and control protocol for both physical and virtual devices
- 4) A baseline policy manager that communicates to the common controller for enforcement
- 5) A mechanism for sharing (communicating or consuming) network state and a unified network state database that collects, at a minimum, MAC and IP address forwarding tables automatically
- 6) Integrated monitoring of overlays and underlays

6. It recommends evaluation as part of the management capabilities, and configuration management that are common for network, compute and storage.
7. It requires open interoperability and programmatic standards for both the control plane and the data plane across solutions.

The working group will list and gather as many requirements as possible from our ONUG community with the idea that the same community votes for the ones that have the highest priority importance to them.

This paper’s goal is to address the function or capability needed from monitoring and management tools, and to remain agnostic on the product, brand or make that satisfy those needs.

Executive Summary

The adoption of open, generic, and even open source solutions has become a general practice. With the explosion of cloud solutions, compute, virtualization and orchestration, the industry has seen the adoption of the virtualization model not only on compute environments, but also extending the concept to storage and network functions, where multiple tenants co-exist, requiring common services that could be either centralized or distributed, and are not only consumed by tenants, but made available to perform a number of actions, including auto improve the process to capture and process data, and analyze it.

More and more, general purpose open operating system environments are being used as a way to replace the custom-made solutions from the past. By the same token, certain areas that were really closed and proprietary environments—for example, the network switching and routing segments—have been experimenting a drastic change with the adoption of commodity hardware chipsets or application-specific integrated circuits (ASICs), which have allowed a model where multiple hardware vendors can be used interchangeably. Now, the open OS and hypervisors are also being ported into these platforms, bringing similarities and commonalities with storage and compute models.

These common services can be a number of different tools and other peripherals necessary for this distributed or local functions that are used for management, monitoring, security, event management, orchestration, patching, analytics, metrics/estate collection, passive and active health-checking, etc., which run as modular applications ([we define each module and function on this link](#)) on these platforms and can be commonly used across compute, network and storage domains.

Common Management Tools Across Network, Storage and Compute Top 10 Requirements

1. Common Management Tools (CMT) implementing interfaces and protocols to guarantee requests/transactions’ integrity, fault management + object state traceability,
2. CMT integration with popular open cloud orchestration solutions, DevOps tools, etc., automation solutions,
3. CMT system utilities/set of native supported features for platform, control plane, statistics, access, performance and analytics management,

4. CMT to collect metrics and telemetry from the operating environment for Storage, Compute and Network,
5. CMT to offer the possibility to add, install or integrate own/third-party applications into the system under the application environment module,
6. CMT systems services module with privileged mode for special programs, applications or critical system functions to guarantee desired Service-Level Agreement (SLA),
7. CMT capability to include a virtualized service or virtualized function platform offering multipurpose common services environments,
8. CMT support for Zero Touch Provisioning (ZTP) and automated provisioning/configuration of storage, network and compute,
9. Integration of complementary/collateral tools to reuse existing solutions, including common tools for event management, telemetry, forensics, self-remediation, traffic captures and security, and,
10. An open certification program to validate, test and certify applications and system services for performance, interoperability and compliance with the common management tools.

Use Cases and Opportunities

1. Generally available, transparent and simplified monitoring of entire infrastructure:
 - Collection of metrics and Time Series Data (querying, pooling, trapping) across any type of assets on the infrastructure, whether they are compute, storage or network systems,
 - Use of standard OS metrics rather than case-by-case isolated management information bases (MIBs),
 - Standardized and simplified correlation engines using similar signatures,
 - Possibility to create common open standards for monitoring across systems and certification programs for application interoperability, and,
 - Elimination of middleware monitoring agents by having the possibility to directly report to distributed big data footprints.
2. Unified management and orchestration of complete infrastructure:
 - Common orchestration and automation for entire infrastructure, and not just the private cloud,
 - Standard service delivery process and improvement for service and catalog management,
 - Ubiquitous tools for management, automation and control of the entire infrastructure,
 - Control plane management and programmatic capability to customize the data plane of every device,
 - Capability to synchronize systems and applications state among multiple regions' standard and unified security enforcement monitoring throughout entire infrastructure,
 - Capability to segment and isolate infrastructure and controlling it on every asset,
 - Unified and standard Authentication, Authorization and Accounting (AAA) across all platforms,

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



**Open Networking
USER GROUP**

- Role-based access control across the entire infrastructure,
 - Standard auditing capabilities, and,
 - Authorization as a service.
3. Ubiquitous support and troubleshooting for all environments:
- Common operational tools for the new infrastructure administrator role rather than the network/compute/storage administrators,
 - Elimination of silos between tools used for compute/storage/network,
 - Capability to run forensics and self remediation, and,
 - Simplification of reporting and capture data by using simplified metadata of any flow or metric within any system.

Conventions

The following conventions are used throughout this document. The requirements that apply to the functionality of this document are specified using the following convention. Items that are **REQUIRED** (contain the words **MUST** or **MUST NOT**) will be labeled as [Rx]. Items that are **RECOMMENDED** (contain the words **SHOULD** or **SHOULD NOT**) will be labeled as [Dy]. Items that are **OPTIONAL** (contain the words **MAY** or **OPTIONAL**) will be labeled as [Oz]. In addition, a priority value of High (H), Medium (M) or Low (L) may be assigned to each item. The priority will be labeled as [RHx], [DHy] or [OHZ] for High priority, [RMx], [DMy] or [OMz] for Medium priority or [RLx], [DLy] or [OLz] for Low priority. The integer values {x, y, z} shall be unique across the document but are not required to be unique across the 3-tuple set {x, y, z}. For example, RM10 and DM10 are allowed whereas RL10 and RL10 are prohibited. Requirements in this document are numbered using increments of 10.

Where needed, related sub-requirements are numbered using increments of 1. The key words “**MUST**,” “**MUST NOT**,” “**REQUIRED**,” “**SHALL**,” “**SHALL NOT**,” “**SHOULD**,” “**SHOULD NOT**,” “**RECOMMENDED**,” “**MAY**” and “**OPTIONAL**” in this document are to be interpreted as described in (Request for Comment) RFC 2119. All key words use upper case, bold text to distinguish them from other uses of the words. Any use of these key words (e.g., may and optional) without [Rx], [Dy] or [Oz] is not normative. The priority assignments are defined as follows:

- **High (H):** Functionality that must be supported on day one and is critical for baseline deployment.
- **Medium (M):** Functionality that must be supported, but is not mandatory for initial baseline deployment.
- **Low (L):** Desired functionality, which should be supported, but can be phased in as part of a longer-term solution evolution.

Problem Statement

The initial issue between these different domains (storage, compute, network) is that, in most cases, each is seen as a completely different practice, and even between professionals that work in each area, there is little collaboration or communication among them with a tendency to create silos. The new environments and challenges derived from the “Everything as a Service” model requires an unprecedented level of integration that lets us predict the future integration of the engineering roles, working all three disciplines into the

integrated role of the infrastructure engineer or administrator, which in turn would also need to integrate extensively a more intimate knowledge of a fourth discipline, “Automation and Orchestration.” Therefore, common tools and unified solutions could be the glue that will help the new infrastructure engineer or operations personnel to bridge the gap between them.

Another challenge is to define common reference architecture for all three disciplines and practices in order to recommend or produce common utilities, services and modules that can be used interchangeably and that would offer the depth and reach that each discipline requires. Based on this, the working group has taken the initiative to define a reference architecture that can be extensible enough to define all of the interfaces and interrelations between systems, and how the common tools and services can be adapted for all of them.

Other issues that offer challenges and would also benefit from a common strategy for management and monitoring are:

- In spite of all the new advances in technology over the years, it is still a struggle to find the right mix of monitoring for devices and servers. Whether polling or trapping or collecting the data, correlating, reporting and performing event management, there are many gaps on how to monitor, what is monitored, and what is really detected.
- Efficiency and effectiveness of monitoring tools offers many challenges. Despite the different monitoring products and technologies that are currently used in an overlapping fashion throughout enterprises, and networks in general, many elements are still not covered. The amount of traffic generated by monitoring solutions may be, in some cases, 30% or more of the total amount of network traffic, and in many cases, there are duplicates or triplicates of the same monitoring, either because it is resent to multiple recipients or multiple solutions monitor and report on the same devices.
- Legacy devices or technologies would not offer the same disaggregated or open model and so, they will need to be dealt separately or may not be integrated.
- In some instances, embedded OS platforms, even when using open OS, may not allow the installation or addition of more applications or complementary tools, which could limit the usefulness of the model. As more powerful processors make their way into bare metal, compute, network switches and other devices, vendors offer full distributions of the OS being ported, given the capability to add a large array of tools and functionality.
- The ability to have modules, along with the capability to add tools, available applications, third-party tools or complementary tools that have not been completely certified or validated by vendors to be used in their OS or devices, may bring issues that require a certification program requiring more overhead to the process.
- Lack of standardized northbound API solutions makes the adoption of specific tools difficult.
- Although there are benefits seen from automation, and orchestration, there is little self-remediation or self-healing capabilities.
- Many complementary applications available are open source. In the past, several disciplines considered open source to be unreliable, but, that approach has to change.

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015

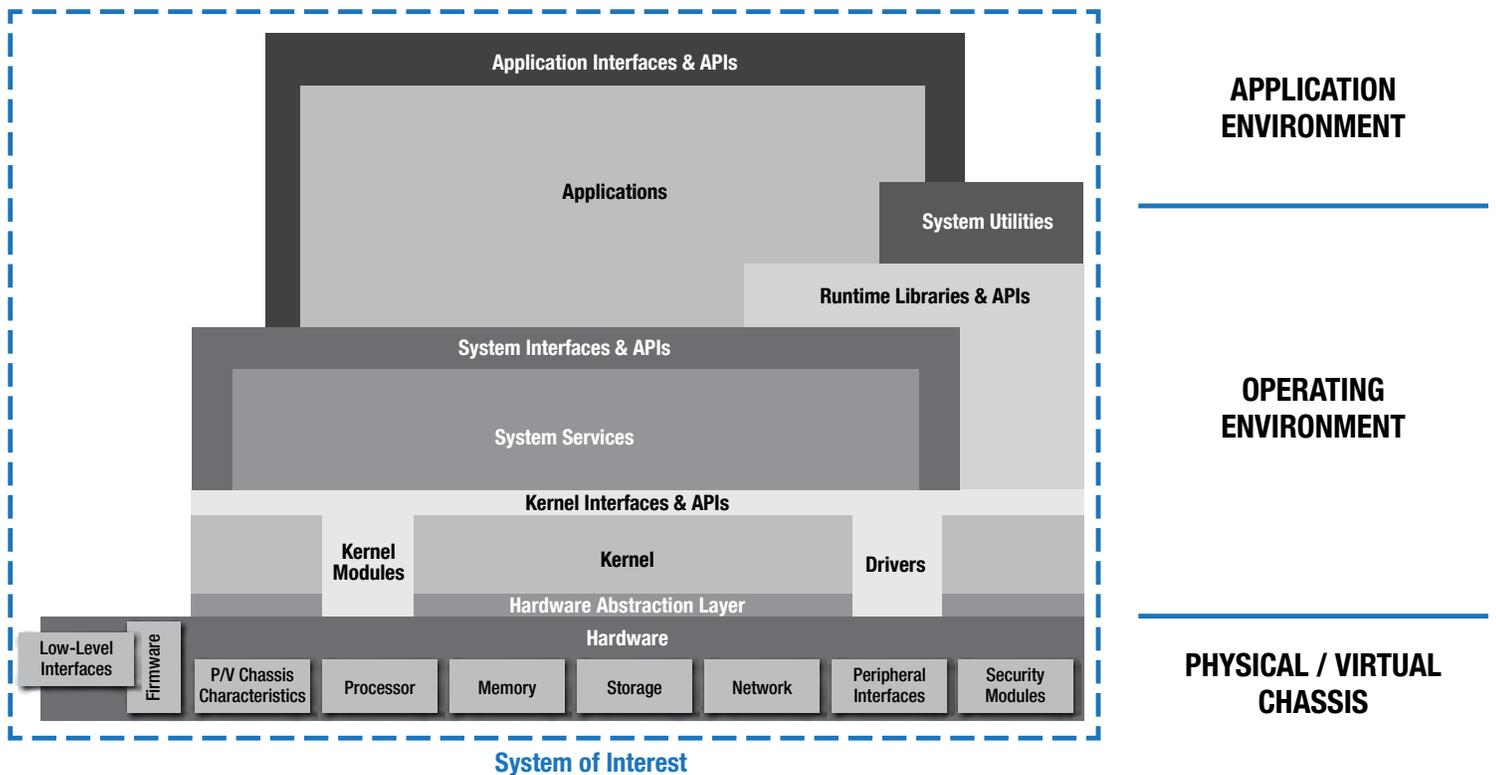


**Open Networking
USER GROUP**

General Reference Architecture for the Common Management/ Monitoring across Storage, Compute, Network Model

To define extensive and open solutions across multiple domains, the group needed a reference architecture that could be used generically for all types of systems, and extensible enough to represent the modules and connectivity points for subsystems and general interfaces. The figure below depicts an alternative view, or model, of modern IT technology. This model comprises three aggregated layers that generically represent:

- Hardware in the form of a physical or virtual chassis;
- An operating environment that can represent traditional operating systems, virtual machine monitors (hypervisors), or embedded system software environments, along with their requisite system services, runtime libraries, system utilities and related software interface (API) definitions;
- Application environment that describes software systems layered on the operating environment and defines the systems' functional purpose or nature (server, workstation, network switch/router, storage appliance) as facilitated through multiple levels of APIs.



These aggregated layers and their respective constituent layers are intended to be both iterative and recursive; that is, elements of the model may be replicated horizontally, or hierarchically instantiated to construct more complex system-of-systems, or decomposed hierarchically into constituent system elements. In all cases, this model does not rely on specific examples of commercial or open systems solution architectures, designs or implementations to describe the features, capabilities or systemic qualities of the system of interest.

The assertion being made here is that all modern IT systems providing compute, network and storage services can be described using the model elements depicted in the figure above and described below. In doing so, the relevant features, capabilities, interfaces and systemic qualities of the desired system will emerge at a sufficient level of detail so as to develop properly formed requirements and definitions (statements of need) that are both measurable and testable.

Physical/Virtual Chassis

The physical/virtual chassis aggregation layer models either a physical hardware system or a virtual system that mimics a physical hardware system. The model is intended to describe the constituent elements of system classes consisting of processing systems, storage systems or networking systems; that is, servers and workstations, network switches and routers, storage arrays and appliances, or the aggregation of these elements into special-purpose devices or the systems-of-systems that define a modern IT environment.

Hardware – This represents the physical or virtual hardware expressed in the form of a “chassis.” All modern IT systems share some or all of the constituent system elements shown in the figure to some varying degree. Every IT system has some form of processor, memory or temporary/volatile storage (although this is now converging with non-volatile storage) and non-volatile storage (data that persists across system invocations), be it magnetic, flash or other semi-conductor-based technology.

In modern systems, every device has some form of network connectivity, or an interface that facilitates communications with external systems and a low-level interface that enables direct communication with the underlying hardware. As such, this model can be used to represent various classes of servers, storage and network systems, in either the physical or virtual world, by varying the number and type of processors, memory, storage, network, peripherals and/or security devices that characterize each discrete system, or expressing those objects as parameterized software abstractions.

Processor – This represents all forms of single-instruction, multiple data (SIMD) and multiple-instruction, multiple-data (MIMD) processing architectures and derived variants. This includes conventional workload CPUs (e.g., x86, scalable processor architecture - SPARC, Advanced RISC Machines - ARM), as well as highly integrated system-on-a-chip (SoC), graphics processing unit (GPUs), general-purpose computing on graphics processing units (GPGPUs) and special-purpose processing devices (field-programmable gate array-FPGAs, demand-side platform - DSPs, custom ASICs). This also includes multi-processor systems (Symmetric multiprocessing - SMP, non-uniform memory access - ccNUMA, NUMA). In the physical world, characteristics of interest include specifics about processor capabilities, processing units (cores), cache levels and sizes, clock speeds, workload efficiency, memory management, integrated subsystems (memory management unit - MMU, peripheral component interconnect (PCI), Universal Serial Bus - USB, network), hardware instruction sets (x86_64, SPARCv9), instruction set extensions for important operations such as virtualization, message switching/routing, encryption and floating-point compute (Advanced Encryption Standard - AES, Intel’s virtualization technology - VT-x, VT-d, [Advanced Vector Extensions](#) - AVX2, Advanced Micro Devices virtualization - AMD-V), power and thermals. In the virtual realm, important characteristics include virtual machine instruction sets (byte-code), processor and socket affinity, peripheral pass-thru (GPU, PCI, USB, storage) and execution caps.

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



**Open Networking
USER GROUP**

Memory – This represents the system’s primary runtime storage subsystem, holding both data and instructions. Memory is closely coupled to processor subsystems and includes data storage technologies/devices that may or may not persist data across system invocations. In the physical world, characteristics such as addressable space, density, access latency, throughput, data/address bus width, error correction and power consumption are of interest. In the virtual domain, addressable space, latency, isolation and oversubscription are of interest.

Storage – This represents the system’s secondary storage subsystems, typically holding system and application software and data in a persistent state that is maintained across system invocations. In the physical world, storage density, throughput, Input/output Operations per Second - IOPS, interfaces, protocols, encryption, deduplication and compression are key operational parameters along with systemic qualities related to redundancy, resiliency, recoverability, security and manageability, and form-factor and power/thermal characteristics. In the virtual scope, disk image formats (virtual machine disk - VMDK, Virtual desktop infrastructure - VDI, Hard Disk Drive - HDD, Virtual Hard Disk - VHD, quantum electrodynamics - QED, QEMU Copy On Write - QCOW), image write modes (dynamic allocation, differential, immutable, write-thru, sharable, multi-attach) and interface emulation are the focus.

Network – This represents all forms of system-to-system communication, including classical networks stacks and topologies (Open Systems Interconnection model - OSI model) and fabrics that implement communication protocols. In the physical domain, this represents infrastructure devices (switches, routers, fabric concentrators), telecom service provider demarks (demarcation points), converged and non-converged device interfaces, wired transmission media, wireless transceivers and their management functions. In the virtual realm, this represents software-defined network (SDN) elements that facilitate connectivity between objects through network function virtualization (NFV), and their protocols and related management functions. With the exception of Layer 1 and Layer 2 protocols, almost all higher layer protocols are implemented in software (one exception being encryption), thereby enabling NFV that facilitates SDN in virtualized systems.

Peripheral Interfaces – This represents all input/output device interfaces to the system, including human interfaces devices (keyboards, pointing devices, displays, haptic devices, PCI, USB, Serial Attached SCSI - SAS/SATA, Small Computer System Interface - SCSI, High-Definition Multimedia Interface - HDMI, display picture - DP).

Security Modules – This represents purpose-built embedded devices (implants) integrated within the system for ensuring the runtime confidentiality, integrity and availability of the system with respect to identification, authentication and authorization.

Firmware – This represents low-level software subsystems specifically designed to manage embedded hardware functions, often used to initiate runtime of higher-order software systems.

Low-Level Interfaces – This represents minimal communication methods between a system or device, and an external system or human, typically using a serial protocol (serial console, Joint Test Action Group (JTAG)).

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



**Open Networking
USER GROUP**

P/V Chassis Characteristics – This represents the characteristic of a physical or virtual chassis. In the case of a physical chassis, this would include characteristics or specifications related to size, mass, construction materials, cooling fans, power supplies and other supporting physical attributes of the chassis. In the case of virtual chassis, this may include characteristics related to how a hypervisor or other virtualization machine monitor emulates devices, arbitrates access and allocates the physical resources of the virtualized host on behalf of the guest operating environment, such as device emulation, processing execution caps, memory oversubscription, peripheral pass thru and the like.

Operating Environment – The operating environment aggregation layer models the system software (at various layers) that facilitates management and use of the underlying physical or virtual hardware. This layer represents the software system elements that enable the physical or virtual hardware platform to fulfill its function. In all cases, the primary function of the operating environment is to manage and arbitrate access to physical resources, or the virtualized abstractions of those resources.

Kernel – At the heart of the operating environment is some form of kernel; a low-level software system with exclusive access to the underlying platform that manages and arbitrates access to physical or virtual hardware resources (processor(s), memory and devices). The kernel typically implements some form of a hardware abstraction layer to facilitate portability across different device architectures. Kernels are designed and implemented using various architectures (monolithic, microkernels, modular kernels, etc.).

Kernel Modules – Represents loadable/pluggable software components used to extend kernel functionality to support some specific feature, capability or device not part of the baseline or core kernel. Not every operating environment employs kernel modules. For classical operating systems, kernel modules are developed to support features and functionality (file systems, encryption, and communications) requiring privileged access to resources (processor, memory, peripherals). Hypervisors employ kernel modules to perform memory address allocations, processor context switches or implement advanced peripheral (device emulation) features on behalf of a guest system.

Drivers – This represents a special class of kernel modules that may or may not be loadable, but can be built (compiled in) with the core kernel software elements. Drivers typically implement interfaces for new hardware or software systems or features (cut-through switching) developed after the baseline kernel.

Kernel APIs – This represents software interfaces that allow system software, utilities or other higher-level systems to programmatically access kernel-managed system resources and services in a controlled and coordinated manner (system calls). These include in-kernel APIs used to manage interaction between kernel subsystems.

System Services – This represents system-level runtime processes integrated within the operating environment that performs higher level functions than those provided by the kernel. These services often run in a privileged mode and are typically responsible for the runtime configuration and management of the system (initialization systems, monitors, load balancing).

System APIs – This represents programmatic software interfaces that facilitate communications between system services and other system services, as well as applications needing runtime support from the system.

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



**Open Networking
USER GROUP**

Runtime Libraries – This represents loadable (memory mapped) object code that provides some generic functionality to system or application processes executable during execution as a callable library routine facilitated through dynamic linking of code segments.

System Utilities – This represents software facilities integrated within an operating environment that perform operational control and management functions to configure and/or maintain the environment according to a defined strategy or policy for use by application processes.

Application Environment – The application environment aggregation layer models the software elements that are deployed on a system to support end users. Typically, this aggregation layer is closely associated with traditional processing, end user oriented systems (desktops, servers, mobile devices), but can be extended to describe the adaptation of network- and storage-centric systems that permit end user customization (appliances, cloud services).

Applications – These represent a class of software processes that perform specific functions or tasks on behalf of an application consumer (end user). Applications may include software processes that run in unprivileged contexts, or those that integrate application functions with system functions through application APIs and system APIs (Type-II hypervisors).

Application APIs – This represents programmatic interfaces to a software application process.

Definition of Common Monitoring and Management Services Modules

The use of closed proprietary system solutions, and closed control planes in the past, made the integration or combination of tools and utilities under different types of systems very difficult. As the use of common operating systems, open solutions, open architectures and interoperability between platforms is being made available across compute/storage and network equipment, that tendency has started to reverse itself.

With the possibility to include standard and common utilities, automation/orchestration, operationally rich systems services, and even with the possibility to add complementary or third party tools, systems implementing the common management and monitoring model can offer a truly open environment that could be characterized as disaggregated services where the power not only comes from the functionality of the main platform's purpose but from the sum of the parts.

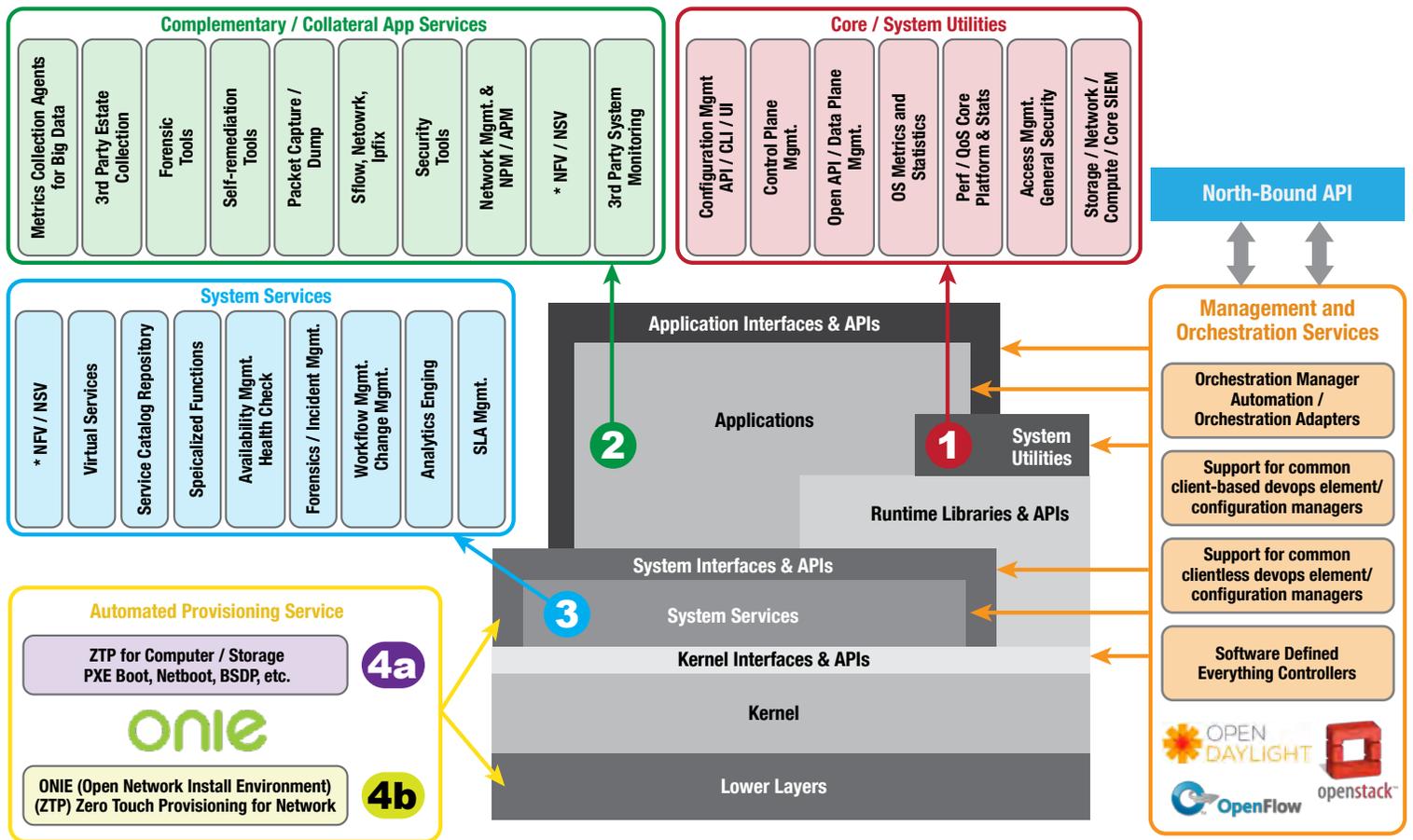
**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



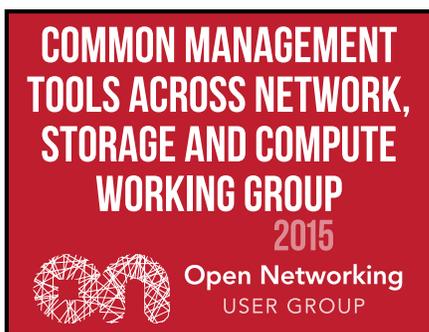
**Open Networking
USER GROUP**

The figure below shows, in general terms, the adaptation of multiple different groups of services to the unified system's reference architecture structure.



As seen in the reference above, there are a number of modular entities in which specific services can be provided or offered to achieve those functions. The vision is not limiting, but inclusive, as the working provides ideas, use cases and examples of what the tools could be or how they can integrate on the different modules of the solution, and many more ideas about how to adapt and integrate all of the functions and features. In general terms, the modules identified above can be categorized in the following groups:

- A) Management and orchestration services to offer integration with both, the more popular open orchestration systems and other well-known DevOps orchestration and automation trends:
 1. System utilities (native or original functionality defined by the creator of the solution, showing general software utilities integrated within the operating environment that perform main operational control and management functions),
 2. Complementary/collateral application services—third-party applications or additional applications—that can be integrated or added into the solution as a way to adapt existing tools that can help improve functionality, adapt to existing requirements, or even a way to reuse/integrate tools and products that have demonstrated their usefulness or compliance to standards and open solutions,
 3. System services (system-level runtime processes integrated within the operating environment that often is running in a privileged mode); in this case, the big difference is that specific functions, products or services can truly be ported with a higher level of control on how the resources of the system could be shared or



be given access to, making it possible to create services virtualization and a great platform for a combined infrastructure as a service Integration. A network Top of Rack (TOR) switch could also offer firewall and load balancing capabilities among others, and many other examples applicable to storage and compute as well, and,

4. The Automated Provisioning Service, ZTP for compute, storage or network that allow for systems to automatically boot, get an address, install the right OS needed and obtain via automation right configuration with minimal intervention.

In the following sections, the potential and use cases, and the requirements that have been defined for each of them, will be discussed in detail.

General Reference Architecture Requirements

[See Appendix A for definitions and disambiguation](#)

RH-10 Each system element within the system-of-interest shall be capable of reporting its current state to an external subject, within the context and control of an integrated or cooperating access control system, implementing a defined access control policy.

RH-20 Each system element within the system-of-interest shall implement an interface and protocol capable of receiving subject requests (queries) for object state data and reporting such data to a subject authorized to receive it. A subject shall be able to request multiple object state elements within a single request.

RH-30 The interface and protocol shall include a method to report an unsuccessful attempt to request object state that provides human interpretable diagnostic messages describing the relevant reasons or causes leading to an unsuccessful attempt, at a sufficient level of detail so as to be actionable by a human subject.

RH-40 Data for each object state element reported by a system element within the system-of-interest shall consist of at least the following data items:

- unique object identifier* - a token or other identifier that uniquely identifies the object;
- object state element label* - an unambiguous label, or tag, identifying the object state element;
- object state element value* - the value assigned to the object state element being reported;
- object state element datatype* - the data type of the object state element value being reported;
- object state element units* - the value units of the object state element being reported;
- object state element mutability* - whether object state element value is mutable or immutable;
- object state element permissions* - an indication of the actions of the requesting subject; and,
- object state element timestamp* - a date/time stamp as described by ISO 8601/ RFC 3339

RH-50 By default, data for applicable state elements reported from an object within the system-of-interest shall be reported using International System of Units (SI Units).

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



Open Networking
USER GROUP

-
- RH-60** Each system element within the system-of-interest shall be capable of receiving requests from external subjects to alter its current state, within the context and control of an integrated or cooperating access control system implementing a defined access control policy.
-
- RH-70** Each system element, upon receiving a request from a subject to alter its state, shall validate the syntax and semantics of the request. If the alternate request is deemed valid, then the object will attempt to satisfy or execute the request. If the alternate request is deemed invalid, then the request is rejected, and the subject is notified of the rejection in a manner that facilitates an actionable response to the rejected request.
-
- RH-80** Each system element, upon receiving a validated request from a subject to alter its state, shall attempt to satisfy or execute the request on behalf of the subject using similar notifications as in RH80.
-
- RH-90** Each system or system element (hardware or software) within the system-of-interest capable of reporting state shall be able to report the following immutable object state elements to an authorized subject:
- uid* - an identifier for the system or system element that is unique within the system-of-interest;
 - name* - a human-interpretable symbol or moniker unique to the system or system element;
 - short description* - a human-readable descriptive summary of the system (one Sentence);
 - long description* - a human-readable descriptive statement of the system or system (one Paragraph);
 - supplier* - the source or originator of the system (manufacturer, vendor or open-source community);
 - creation date* - the supplier provided date/time when the system or system element was manufactured or otherwise created; and,
 - install date* - the date/time when the system or system element was introduced.
-
- RH-100** Hardware systems and system elements capable of reporting state shall report the following additional immutable object state elements to an authorized subject:
- model* - the supplier designation for the system or product;
 - revision* - the supplier designated engineering change version for the system;
 - serial number* - the supplier designated unique identifier for the system;
 - parent* - the system for which this system is a system element; and,
 - power-on hours* - the accumulated number of hours since installation.
-
- RH-110** Software systems and system elements capable of reporting state shall report the following additional immutable object state elements to an authorized subject:
- version* - the supplier designated software revision for the system;
 - release* - the supplier designated software version variant provided to a consumer;
 - install location* - the base install path where the software system element is installed;

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



Open Networking
USER GROUP

software group - the family of software systems to which this software belongs;
software license - the license under which the software is released;
package filename - the container file holding all of the software;
package size - the uncompressed size in bytes of all software elements in the package;
package signature - the cryptographic hash/key, signing date and hash algorithm used to verify; and,
package manifest - the contents of the software package.

Services Modules, Services Definition and Requirements

The idea of disaggregated services is not only applicable to the services modules, but also as follows: as a general note, the use of an OS (Linux, Windows, Solaris, Mac OS X, AIX, FreeBSD, NetBSD, and OpenBSD or common hypervisors ESXi, XEN, KVM, etc.) for storage, network devices or compute brings the capability from each case that all of the same tools and applications that can run on those common OSs can be used by all three disciplines uniformly. The concept is adaptable to both physical and/or virtual devices/appliances/servers.

Orchestration and Automation Access Points

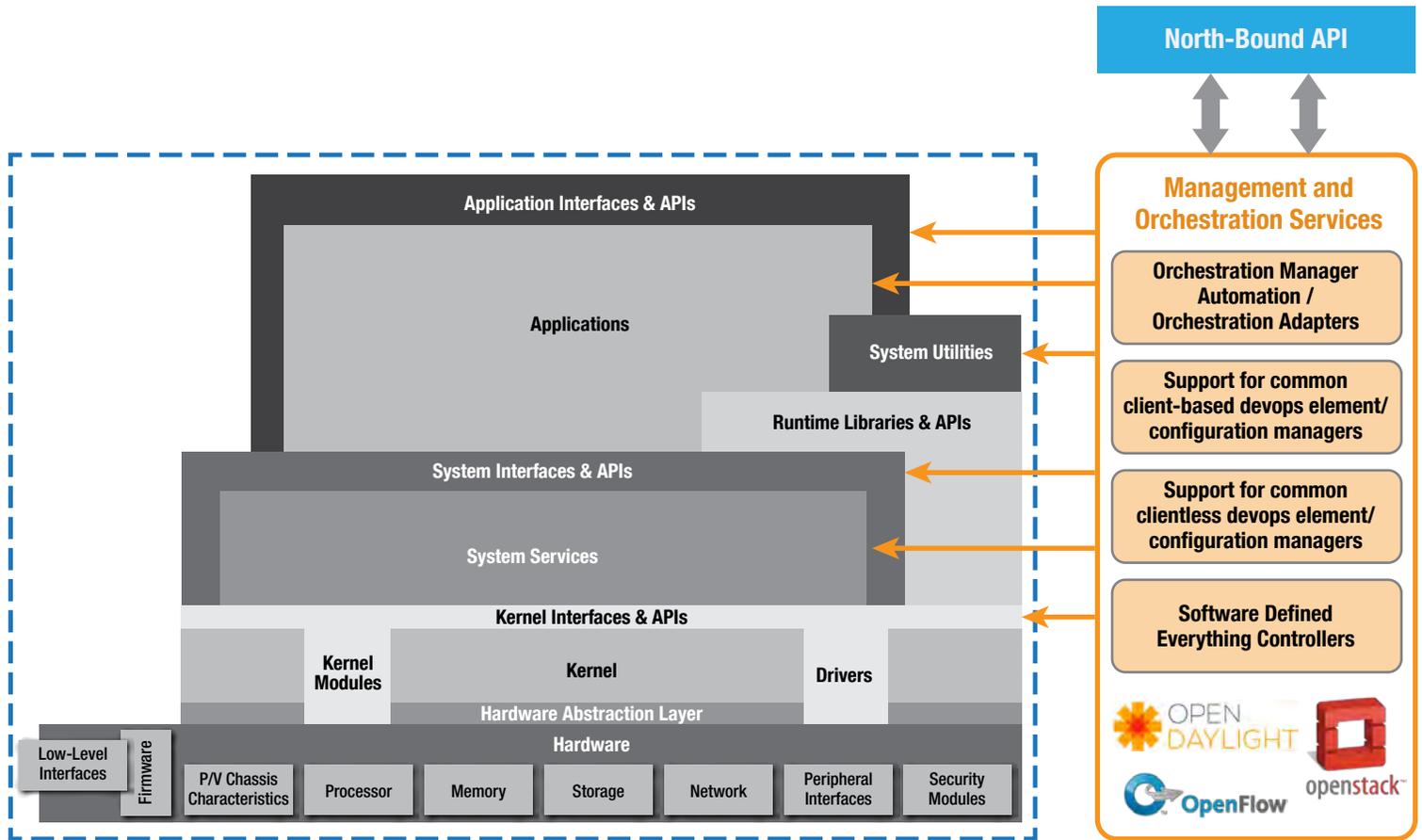
As the working group has kept a vendor- and product-agnostic focus, it distinguished the level of integration, interrelations and interactions that automation and orchestration systems have with several open source projects, and other so-called [DevOps tools](#), which the group has tried to leverage and integrate into the vision instead of trying to redefine or reinvent them. The working group acknowledges the most commonly used tools for orchestration and management.

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



Open Networking
USER GROUP



On the other hand, it is important to recognize that these orchestration solutions, whether outsourced or built within an organization, could have multiple levels on which they will interact with the reference architecture depending on the type of system that is being provisioned or the function being automated such as:

- Own orchestration manager or automation engine containing the northbound API where users will make requests. Containing or connecting to automation adapters, element managers/configuration managers, automation adapters or element/configuration managers that control and execute the automation internally;
- [DevOps orchestration](#) solutions that may use a client or agent-based model requiring the agent to be installed on the system and connected securely to an orchestration server executing the desired flows;
- [DevOps orchestration](#) solutions that may not use a client or agentless model requiring the system to be securely connected to an orchestration server executing the desired flows;
- Integration with third-party “Software Define Everything” (SDE) controllers via plugins or other integration points that may control the system as a whole or a portion of its operations.

As some of the common open management and orchestration solutions, like [OpenStack](#), [OpenDaylight](#), [OpenFlow](#), are leveraged, there is an understanding that the automation capabilities will not only be able to use and interface with [DevOps clients](#), but that depending on the function, there would be support for some of the common protocols for configuration and management like NETCONF, Open vSwitch Database Management



Protocol (OVSDB), a programming language (LISP), Yet Another Next Generation (YANG), data plane programmability ([See Core System Utilities section](#)), etc.

Orchestration and Automation Requirements

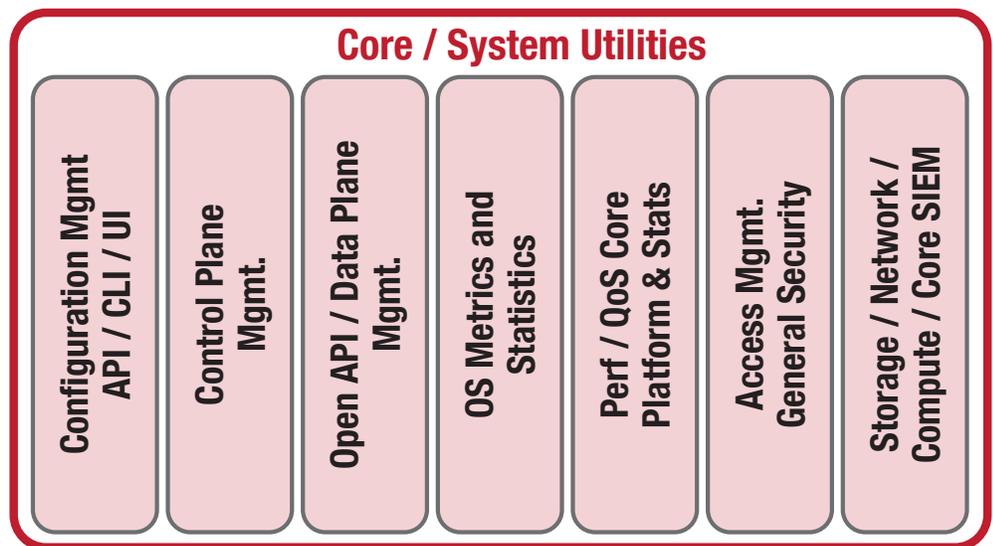
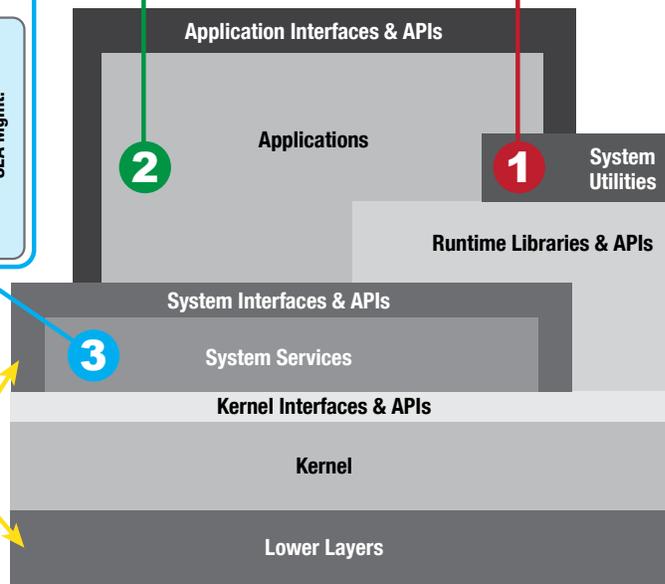
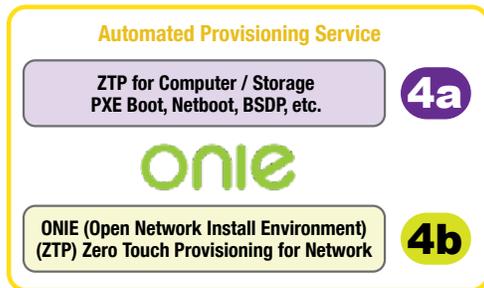
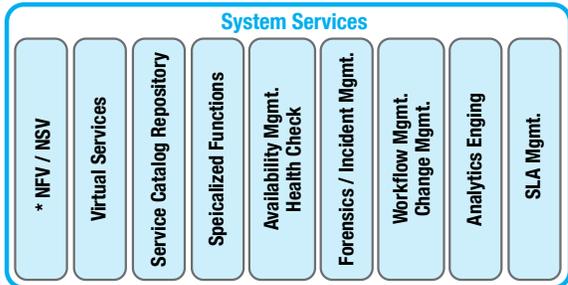
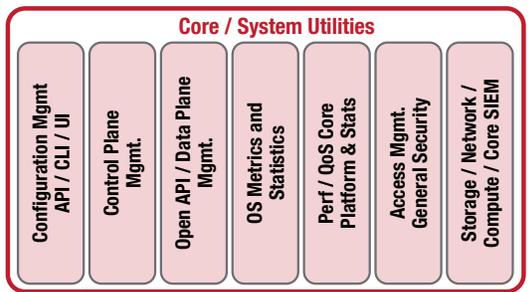
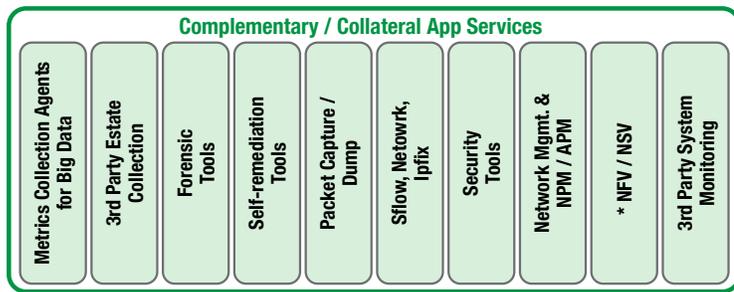
- DH-10** Solutions or platforms looking to be compliant with the common management and monitoring systems model should have support and integrate with most common, popular orchestration and automation solutions with a preference, but not limiting for open standards and open platforms,
-
- DH-11** Should have support and integrate with open cloud orchestration, management and automation solutions (e.g. OpenStack),
-
- DH-12** Should have support and integrate with most popular [DevOps](#) orchestration agent or agentless solutions for both provisioning and other frequent change management automation and orchestration,
-
- OH-10** May include support and integrate with SDE, depending on the case, understanding that depending on the type of system and scope may or may not be an open software-defined framework for the use case, but the optional requirement is used on a flexible progressive way as new Open Defined Solutions-proprietary and non-proprietary continue to appear (e.g., OpenFlow, OpenDaylight, software-defined storage commercial products, SDN commercial products, etc.),
-
- OH-20** As a requirement for users and not for developers or vendors, users shall conduct a complete strategic evaluation and position recommendation on how automation and orchestration would be utilized in their organization, and what short- and long-term plans would be implemented, which tools would be used to automate the provisioning and change management capabilities for storage/ compute, and network, including roadmap, tools and feasibility analysis to map all that criteria to possibly integrate with this model.
-
- RH-120** Connectivity between orchestration engine, clients and the system being controlled must be secured requiring elements of confidentiality, integrity and authorization to gain access.
-

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



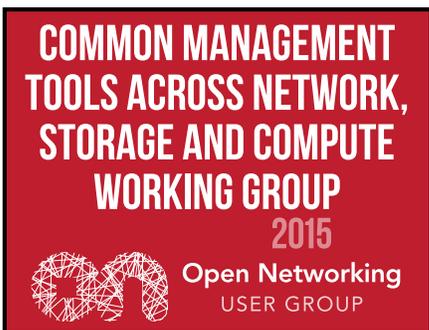
Open Networking
USER GROUP



Core System Utilities

As explained before, system utilities has been defined as the software facilities integrated within an operating environment that perform operational control and management functions to configure and/or maintain the environment, according to a defined strategy or policy for use by application processes.

In other words, the working group refers to these as the tools that the vendor or organization creating the platform would include as part of the solution to manage and control it. They may be thought as proprietary, but they could also be part of a new



standard which all vendors could use openly and be as open as it could be. Including on this are, but not limited:

- UI/API/CLI that will allow to manage the solution and access to the system,
- Control plane management and programmatic API in the event that the solution is a SDE where a specific controller will manage the control plane directly,
- Possible integration point with open data plane programmatic capabilities or standards that can program the data plane on different types of systems (as an example of possible open solutions mapping to this space, the Open Compute Project Switch Abstraction Interface (SAI) and other initiatives that are being used to program the data plane of devices,
- OS metrics, statistics and reporting that will help obtain metrics and collect any possible value regarding performance of the system, including any information regarding low-level functions on the reference architectures. Moreover, there could be a standard list of metrics that will be required for storage, network and compute that should be available from every system implementing a common monitoring and management model,
- Access management, role-based access control capabilities, capability to enforce segmentation and isolation, full audit capabilities and authorization as a service for permissions that are required on a per-session basis,
- Performance/QOS metrics, throughput, bandwidth, utilization, telemetry, analytics and statistics of core platform; moreover, there could be a standard list of metrics that will be required for storage, network and compute that should be available from every system implementing a common monitoring and management model,
- Possible native support for Time Series reporting application for all of the systems' metrics that can be reported to a distributed database or big data solution,
- Core functionality operations and management (Syslog, Simple Network Management Protocol (SNMP), event management, Traps, NetFlow, sFlow, etc., that could be pooled, trapped or constantly streamed as metadata).

Core System Utilities Requirements

DH-20 Systems implementing the common management and monitoring model should offer a well-defined set of system utilities with a set of core features, which would map into the systems utilities' services module of the [Reference Architecture](#). The utilities in this section could also be standardized and be available as part of open solutions and standards looking to promote common management/monitoring tools across storage/compute/network. As an example of some possible utilities:

- Configuration management API/command line interface (CLI)/user interface (UI),
- Control plane management,
- Open APIs/data plane management,
- OS metrics and statistics,
- Performance/QOS core platform statistics,
- Access management general security, and,
- Storage/network compute/core security information and event management (SIEM).

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



Open Networking
USER GROUP

RH-130 The platform or system must offer an API and capability to request any action that can be executed, including metrics collection on the platform through the API. In addition, standardized and common front side APIs for compute, storage and network devices/servers should be offered for all systems and solutions.

RH-140 Capability to collect metrics and telemetry from every aspect of the OS features and properties, depending on the type of device or system, and multiple metrics must be made available interchangeably as part of the packages that should cover storage, compute and network.

- Storage: Collection of metrics related to storage should include elements like input/output (I/O) operations, response time for operations to be completed, bandwidth being moved, storage interfaces utilization, space availability, reads per second, writes per second, read rate, write rate, usage average, read latency, write latency, disk command latency and commands per second.
 - Compute: CPU, memory consumption, high process CPU %, load averages, resource allocation, total number of VMs, total number of data stores, number of VCPUs powered on VMs, workload indicator and average running VM.
 - Network: CPU, memory used, switch port/interface availability, interface utilization, memory utilization, utilization, QoS metrics, queues size and buffers sizes, drop packets, jitter, routing statistics, maximum throughput, etc.
-

DH-30 Should offer the capability to allow a controller or through the API to the control plane, or other mechanisms to be able to provision all new settings, make changes, add, update or delete settings or previous configuration.

RH-150 System must support role-based access control integrated security for automation API where users are controlled on what they can or cannot request, and, hopefully, can be implemented with a security-driven solution implementing authorization as a service.

DH-40 Systems implementing the common management and monitoring model should offer the capability to allow connectivity/access to the control plane management element of the system to a controller or controlling entity to execute on-demand monitoring and configuration management.

DH-50 A common monitoring and management server/appliance should allow programmatic control of the data plane of the system with open standards, like Open compute Project's Switch Abstraction Layer ([OCP's SAI](#)), [Broadcom Network Switch Library \(NSL\)](#), Intel Pentium 4([P4](#)) and other open protocols, to be able to program different environments and multipurpose devices making it adaptable to many different ecosystems and conditions to process multiple types of tasks.

DH-60 Systems implementing the common management and monitoring model shall be able to use and enforce authorization as a service based on a role-based access control's work flows to apply policy and permissions of who can access the

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

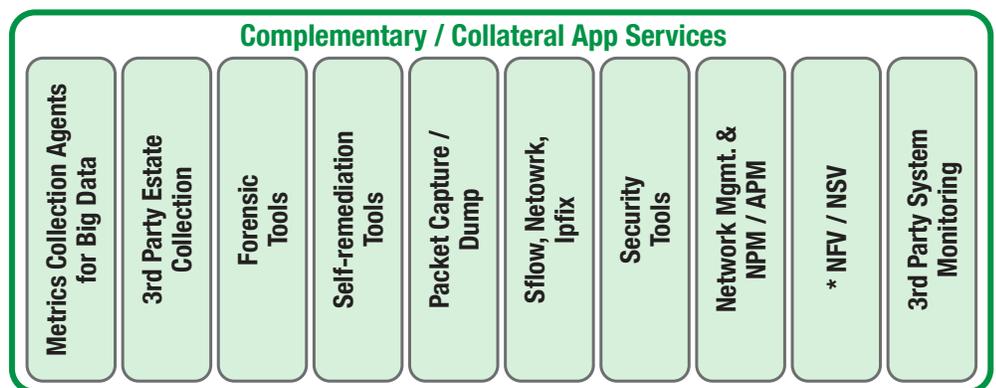
2015



Open Networking
USER GROUP

different applications within the device/server.

- DH-70** Systems implementing the common management and monitoring model should be capable of offering a standard AAA (Authentication, Authorization, Accounting) across all platforms, and role-based access control across the entire infrastructure. (The problem today is that storage and compute use Lightweight Directory Access Protocol (LDAP); network uses Terminal Access Controller Access Control System (TACACs)/ RADIUS).
- DH-80** The solution should offer complete auditing capabilities of actions executed by every user and any changes or updates on the system.
- DH-90** Systems implementing the common management and monitoring model should have the capability to enable event management via Syslog, SNMP polling/Traps, NetFlow, sFlow, to either a centralized agent or to a distributed aggregator.
- DH-100** Systems implementing the common management and monitoring model should have the capability to comply with SIEM systems, by being able to generate alarms and notifications for condition problems and the capability to correlate those alerts with specific actions and specific level of awareness.



Complementary/Collateral Application Services

On this module, there are application services that can be integrated or added into the solution as a way to adapt existing tools that can help improve functionality, complement operational support, troubleshooting, adapt to existing requirements, or even a way to reuse/integrate tools and products that have demonstrated their usefulness or compliance to standards and open solutions.

This category could be used for additional applications that could be added to the solution as a separate installation on the OS or a possible option that the vendor may allow for applications on the platform (app-store like of capability); could include open source or proprietary tools but, not limited to these.

The difference between these applications/utilities and the system services is that the system services can be set to run as privileged mode and assigned priority, and resources which in the case of complementary collateral application may be a best-effort solution unless an SLA can be setup with the provider of the system or solution.

Some vendors creating or developing systems may select to include an existing well-established or proven tool as one of these functions instead of creating them, given how

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



Open Networking
USER GROUP

mature some of those features and software are. By the same token, an enterprise can select to install the same application on all storage/compute/network domains, and pull or push any information.

Note: In order to stay vendor/product-agnostic, none of the names of applications or vendors that may apply to this category are included, and a number of open applications in different categories are mentioned; but, in reality, the analogy of the app store applies here, as an application is available for any needed tool or one can be created.

- Metric collection and reporting of Time Series to big-data collectors; in the future, many monitoring agents could send data or metadata directly to Hadoop or other big-data solutions, and even with present applications, like OpenTSDB as an example, that can do this to send Time Series Data for many metrics collected in keeping a distributed database.
- Forensics, verification and self-remediation tools; with programmatic capabilities like SDN and other automation tools, monitoring applications could gather forensics for a specific action or event, and trigger or execute actions to either self-remediate or mitigate among many other actions that could execute.
- Third-party agents, agents that may execute any specific collection or task on the platform to push data to any external platform for collection or any internal process or application that can be running analytics on the data collected.
- Packet-capture tools that can sniff traffic from either physical interfaces or virtual interfaces on-demand and store data for troubleshooting purposes;
- sFlow, NetFlow, Internet Protocol Flow Information Export (IPFix), SNMP, Syslog and trapping clients that can execute all of these protocols and send data to multiple receivers.
- Security applications and tools ([these can be used interchangeably with the Systems Services' module](#))
 - Software compliance and host check validation,
 - Security patches and vulnerability testing,
 - Security automation for policy validation,
 - Distributed firewalls and multi-tenancy policy enforcement,
 - Isolation compliance validation,
 - Internal auditing automation,
 - Boundaries and use case compliance requiring separation between environments such as Dev, test, QA, Prod, etc., and,
 - Creation of quarantine zones.
- Third-party monitoring and metrics collection including network performance management and application performance management tools from multiple vendors in the market as applications.
- Service virtualization/NFV. With the increased capacity on systems and more powerful processors, there is the possibility to install a complete virtualized image of a service, and appliance as a complementary application that can serve a storage, compute or network function. Any type of virtualized function could be set as another application that could be used for any purpose.

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



**Open Networking
USER GROUP**

- The difference of setting NFV or SFV at this level or at the system services level is that as a collateral or complementary application, it cannot be defined as the priority or resources that would be used for the service or function, and may run as a best-effort service.

As a general comment, in the past, monitoring solutions on the compute and storage sides have shown a high dependence on syslog or other polling mechanisms, while on the network side, there is a high dependence on syslog, and SNMP agents for trapping and polling. With the proliferation of APIs and the use on main streams of big data technologies, there is a move to enable much more advanced distributed and scalable metric collection from each device or server with new capabilities to extract/post metrics, and possible reports of any type of event management directly into big data with no intermediaries or middleware that can execute self-remediation or self-healing by activating automated and programmatic functions.

Independent on whether this application module could allow multiple different types of applications, there is an opportunity to create an open certifiable and testable program where applications can be certified to work with open systems on the application module.

Complementary/Collateral Application Services Requirements

RH-160 Systems implementing the common systems management and monitoring model must offer to vendors, developers or any possible user of the system, the possibility to add, install or integrate their own or third-party applications into the system under the application environment module or segment module specified on the reference architecture. These applications may be used as complementary add-ons to the solution or as part of the systems features supported.

OH-30 A certification program could be established to validate, test and certify that applications and tools would comply with the common systems management and monitoring model to:

- Verify interoperability with other applications,
- Performance and compliance (amount of resources required that can consume or expect to consume), and,
- Integration with common systems management and monitoring general standard API.

DH-110 The following use cases should be in high demand for, and highly usable complementary/collateral applications integrating with the common management and monitoring model.

DH-111 Should offer the capability to trigger on-demand verifications and gathering forensics.

DH-112 Should offer event management and correlation capabilities, which could trigger API or other requests to programmatic agents that can execute self-remediation.

DH-113 Should be able to execute data collections and traffic captures that could upload to a big data collector as metadata or some lightweight protocol to obtain analytics.

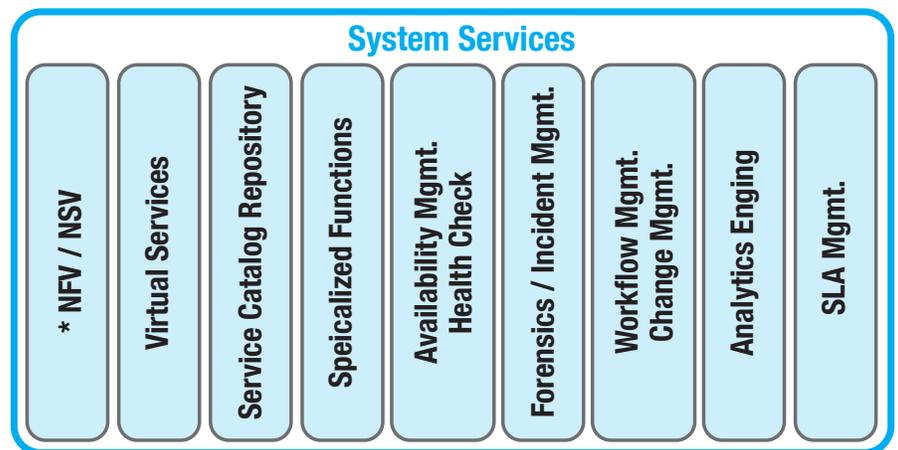
**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



**Open Networking
USER GROUP**

- DH-114** Should be able to perform auditing, vulnerability penetration testing and other security tasks.
- DH-115** Should have the capability to execute policy-based routing, re-directions, dynamic classification of traffic and applications.
- DH-120** Systems implementing the common management and monitoring model could perform application transaction validation and monitoring across environments (web, application, database), monitoring performance of each segment while all information could be send to the correlation or big-data agent.
- DH-130** Systems implementing the common management and monitoring model applications and modules could be created and installed on any open network/ compute/storage disaggregated server/device to validate many different security requirements:
- Software compliance and host check validation,
 - Security patches and vulnerability testing,
 - Security automation for policy validation,
 - Distributed firewalls and multi-tenancy policy enforcement,
 - Isolation compliance validation,
 - Internal auditing automation
 - Boundaries and use case compliance requiring separation between environments such as Dev, test, QA, Prod, etc., and,
 - Creation of quarantine zones.
- RH-170** There should be a maximum amount of resources used within the disaggregated device or server that would be dedicated to complementary applications that are run for monitoring or management purposes on top of the platform.



System Services

This represents system-level runtime processes integrated within the operating environment that performs higher-level functions than those provided by the kernel. These services often run in a privileged mode and are typically responsible for the runtime configuration and management of the system (systems, monitors, load balancing, etc.).

In this case, the big difference with the collateral/complementary services is that specific functions, products or services can truly be ported with higher level of control on the resources of the system to be shared or given access to, making it possible to create services virtualization, and a great platform for a combined infrastructure as a service Integration. As an example, a network TOR switch could also offer firewall and load-balancing capabilities among others, and many other examples applicable to storage and compute as well.

Some of the use cases for system services are:

- NFV may seem oriented to networking, but service virtualization can be any function that could be related to compute or storage as well. In the case of NFV, this is a good use case to demonstrate disaggregation of services where, with the new capabilities of bare metal switches offering more powerful processors, the network IT could install a network function, like a L3 router, firewall or load balancer, or multiple of these, on top of a TOR switch and assign the right level of resources to run these functions on privileged mode to obtain an SLA similar to what a typical standalone device would have.
- Virtual services, similar to the previous case, other services related to compute or storage or certain peripheral services could be given a higher priority than if they were collateral applications.
- Service Catalog Repository, depending on how the services can be implemented either on a centralized or distributed way, the catalog of services can be contained into the systems services area so that the SLA of the services, and the correct amount of resources, can be given to them deterministically.
- For specialized functions, on this case, vendors or open systems creators could build/customize or develop elements of the solutions as processes, daemons, functions or modules that could execute key or signature functions for the solution to work, analogous to the quintessential “secret sauce” of the solution.
- Availability management is to be used by the solution to perform any type of health checking between the parts of the solution or even synchronization between them.
- Incident management would be a specific functionality or module created to determine problems within the solution and a way to collect forensics, troubleshoot or gain intelligence on issues, and perhaps repair implementation.
- Change management would be a specific functionality to allow for work-flow management, audits and visualization of any changes that have been done in the system, and even how they can back out of a change or backup configuration or execute any changes to the system through this module.
- Analytics engine can use any of the metrics provided by the other modules, and services made available to build analytics and business intelligence out of the system, which may include dashboards and other capabilities to perform mining, filter and display estate allowing polling of the data.
- As the Service Catalog Repository or any application can be mapped as either existing on the collateral application service or on the system service module, the system would be able to determine the performance of the application and map it to the policy established to understand the SLA balance.

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



**Open Networking
USER GROUP**

System Services Requirements

- DH-140** Solutions implementing the common management and monitoring tools model should offer a systems services module as defined by the [Reference Architecture](#) where special programs, applications or functions can have elevated control over the possible resources that could be assigned to the system service, in order to offer and support critical or essential programs for the systems' functionalities.
-
- RH-180** System services should have the capability to monitor application performance, and estate, and enforce SLAs for multiple tenants that could have a presence in the device.
-
- DH-150** System services reporting mechanism should continuously report Time Series or other metrics as metadata or continuous information could replace other mechanisms that have not been very efficient over time to detect time-sensitive thresholds proactively.
-
- DH-160** System services could perform health-checking process status, and synchronization of state, and offer that feature as a service to possibly other functions or virtualized services that can be integrated into the solution for proactive and reactive state reporting (pooling/trapping).
-
- RH-190** Solutions implementing the common management and monitoring tools model must be able to install or allocate a platform that can be either a virtualized service or a virtualized function into the system service allocation and assign the resources necessary. That way, any possible server or device can become a multipurpose device that can adapt to multi-tenant and common services environments.
-
- RH-200** The service catalog repository could be, however, part of the complementary applications services module, in order to keep the SLA that the catalog would need, if the catalog of services, or a part of it, will be hosted or contained within the solution, the systems services module must guarantee that the SLA and performance needs of the catalog or parts of the catalog are met.
-
- RH-210** Solutions implementing the common management and monitoring tools model should offer automated collection and feedback of metrics from incident management systems, which could also have direct agents or clients on the devices to collect and trigger automated steps to determine root cause and fix problems.
-
- DH-170** Applications or agents should be adapted from specific vendors or generic ones for the following cases, creating a strong service delivery and service level management capability:
- Interoperability and integration with capacity planning tools and capability to pool and query capacity,
 - Inventory and asset management including Configuration Management Database (CMDB) specific agents,
 - Integrated and interoperability with helpdesk to do automated verifications,
 - Disaster recovery tools or agents that could be configured to execute a number of steps or checks,
 - Security validation and vulnerability testing,

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015

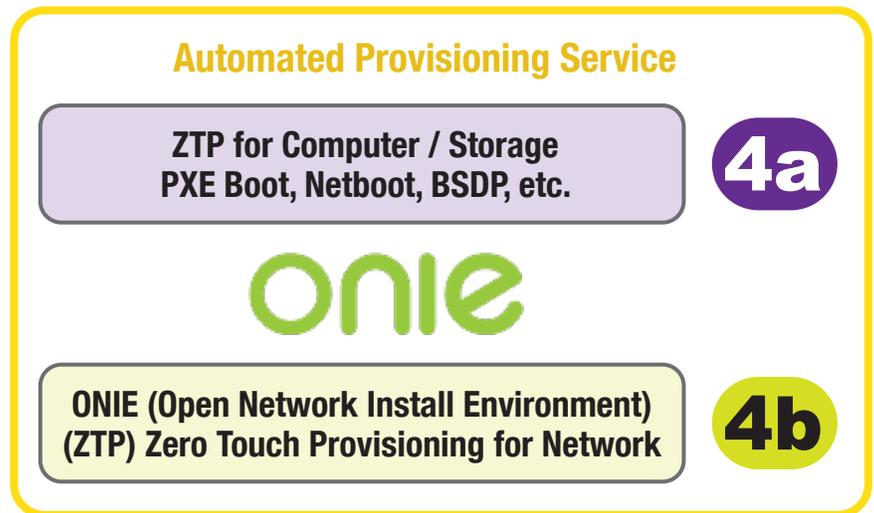


Open Networking
USER GROUP

- Compliance checks and validation of systems requirements versus real, and,
- Security on-demand as host-based checks could be performed.

RH-220 System services module must allow for the creation of general policy and enforcement of contracted SLAs.

RH-230 System services module must allow for the creation or integration of an analytics engine, which shall integrate with other general reporting capabilities, to offer general intelligence to the solution.



Automated Provisioning Service

The Automated Provisioning Service, the ZTP for compute, storage or network solutions, that would allow systems to automatically or via automation and with minimal manual/human intervention, to go through the complete configuration of a system, passing through the steps of initialization or boot process of the system obtaining an IP address, identifying the role that the system will play, installing a required OS, and obtaining and installing the right configuration.

Typically, compute and storage systems have used methods like Preboot eXecution Environment (PXE) boot, netboot, Boot Service Discovery Protocol (BSDP) and others to boot and configure systems automatically. In the case of network systems, even when some parts of the process have been automated, it requires a high percentage of intervention from the operator. Recently, Open Compute Project (OCP) released an open initiative named Open Network Install Environment (ONIE) ([link](#)), which defines a standard way to automate the process of booting and configuring a network device with minimal intervention.

As an opportunity, there could be a possible addition to the standards or a creation of a new standard where solutions implementing the common management and monitoring tools model could include a process to add specific services modules' components and configuration.

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



Open Networking
USER GROUP

Automated Provisioning Service Requirements

- R-240** Systems implementing the common management and monitoring tools model must offer support for ZTP and automated provisioning/configuration of storage, network and compute servers/devices, and the capability to deploy specific configurations automatically into the systems.
-
- R-241** ZTP should offer a way to adapt the preliminary logistics processes so that a simple import would allow for the initial processes, like system registration and asset management.
-
- R-242** Process should allow for conditional programming, depending on the type of system and the segment location within the infrastructure where it belongs.

ONUG Common Management Tools Across Network, Storage and Compute Working Group Members

Carlos Matos



Chairman

Brian Hedstrom



Thom Schoeffling



**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



Open Networking
USER GROUP

Appendix A: Definitions and Clarification

The expression of general or specific requirements in this document may be articulated using object-state modeling. Object-state modeling is a technique that enables systems architects and engineers to describe features, capabilities and systemic qualities that are desired in a system. These needs are expressed along with any constraints that may be imposed by technology, resources, and/or policies governing the development and operational use of the system. The technique is predicated on the notion that abstract representations of systems and their constituent system elements can be used to model system behavior and interaction with actors (humans and other systems), thereby exposing emergent features, properties and interfaces that are relevant to their function and behavior. This can help with developing requirements that are articulated in a manner that is understandable, meaningful and testable.

For the purposes of this document, the following definitions and descriptions are introduced:

object - An abstract representation of a tangible or intangible thing. At a base level, an object can represent a system-of-interest. At a micro level, an object can represent a system element or constituent component comprising the system-of-interest. At a macro level, an object can represent a system-of-systems that has relationship to the system-of-interest on a broader scale. An object can also represent data and its derivatives (information, knowledge, intelligence). Objects exist in both the physical (tangible) and logical/virtual (intangible) domains.

subject - An abstract representation of an actor (human, machine or other entity operating with intent) that has some relationship with an object. For systems, this is typically characterized by an access relationship between the subject and object-where the object provides some service to, or on behalf of, the subject. When subjects interact with objects, they affect the state of the object in some way.

object state - A representation of an object's condition at a given point in time. Every object has state and every system of any consequence has many states, including those systems designated as "stateless." Object state may change in response to inputs introduced to an object by subjects.

object state element - A data element representing some part of an object state as reported by the object. An object's state may be defined by one or more object state elements (a collection of state elements). An object state element is expressed in terms of a parameter with an explicit value, or in the form of a symbol or expression, that represents some collective value derived from multiple parameters (online, offline, restarting, nominal, degraded, fault, panic, overload, hot, full, and saturated).

object initial state - The condition of an object representing the starting point in a continuum of conditions experienced by the object over time. Every object has an initial state. An object's initial state may or may not be its default state. The initial state may be relative to a specific context or perspective of the object that establishes a baseline starting point representing a defined condition along a continuum of conditions bounded by two endpoints in time. An object's initial state may be preceded by a meta-state that describes the non-existence of the object (i.e., prior to creation).

object current state - The condition of an object at a given reference point in time.

COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP

2015



Open Networking
USER GROUP

object state transition - A change in an object's current condition (current state) to another condition (transition state) experienced during a defined period of time (switchport up -> switchport down, RAID re-sync in-progress -> RAID re-sync completed, server shutdown initiated -> server halted).

object terminal state - The condition of an object representing the ending point in a continuum of conditions experienced by the object over time. Every object has a terminal state (although the terminal state may never be reached). The terminal state may be relative to a specific context or perspective that establishes an ending point representing a defined condition along a continuum of conditions bounded by two endpoints in time. An object's terminal state may be followed by a meta-state that describes the non-existence of the object (i.e., after destruction)

system - A combination of interacting elements (physical and/or logical objects) organized to achieve one or more stated purposes. Within the scope of consideration, such elements represent the system-of-interest (server, network switch, storage appliance).

system element - A constituent member (object) or component of a system (e.g., processor, switchport, disk).

system-of-systems - A system whose system elements are themselves systems (e.g., datacenter, cluster, server farm, LAMP stack).

interface - A shared physical or logical boundary between two or more objects established to facilitate communication, cooperation and/or collaboration between objects.

protocol - An agreed upon set of rules and methods governing the exchange of data and/or information between objects in order to realize an interface.

With these definitions in mind, the following assertions can be made:

Any management framework requires that somebody (a subject) be able to affect change (transition) against some thing (an object).

In order for the change to be deterministic, the current condition (current state) of the object needs to be known to some finite resolution by the subject.

The object needs to extend some method and interface in order for the subject to learn or measure the object's current state such that the inputs needed to affect the desired change can be determined.

The object needs to extend some method and interface in order for the subject to introduce inputs that will induce the object to respond in a way that changes its state to the condition(s) desired by the subject.

In other words, for any given object, methods and interfaces are needed to read its current state and alter (or transition) its current state to another state. Metaphorically, these can be thought of as the "dials" and "knobs" of a control panel that someone uses to manage a system. As such, when requirements are developed for any system, the following questions need to be considered:

What subjects (administrators, provisioning systems, management systems) will be initiating changes against a system (an object)?

What objects (servers, networks, storage) will be considered "in scope" for change (the system-of-interest)?

**COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP**

2015



**Open Networking
USER GROUP**

What are the legal states the object(s) can have and what is needed to facilitate transitions between each legal state?

What is needed in a system to respond to attempts by a subject to transition an object to an illegal state?

What is the expected or likely current state(s) of an object at the time a change is initiated?

What are the changes (state transitions) a subject is permitted to affect against a given object or set of objects?

What information ("dials") about the object's state is needed by the subject prior to, and after, initiating the change?

What methods ("knobs") will be used to introduce inputs to the object that will initiate the change?

What capabilities are needed to respond to exceptions, faults and failures experienced by an object?

From this, a set of primitive operations that a subject can apply against an object can be described, and from which more sophisticated operations can be derived:

create - the object is instantiated by the subject; transitioning from the meta-state of non-existence to its initial state;

read - the subject observes or interrogates (queries) the object to learn or measure its current state;

modify - the subject introduces inputs to the object that causes it to react in a way that changes its state;

destroy - the object is terminated (or deleted) by the subject; transitioning from its terminal state to the meta-state of non-existence.

With the foundation described above, general and specific requirements can be articulated for systems providing compute, network and storage services within an IT infrastructure.

Appendix B References

DevOps definition from Gartner <http://www.gartner.com/it-glossary/devops>

DevOps Tools: Loose term used to reference multiple different element managers and configuration managers products used to orchestrate and automate tasks like Opscode Chef / Puppet, Ansible, CF Engine, Dockers, etc

Opendaylight Project, <http://www.opendaylight.org/>

OpenStack Project, <http://www.openstack.org/>

OpenFlow, <https://www.opennetworking.org/sdn-resources/openflow>

Microsoft (SAI), http://azure.microsoft.com/blog/2015/03/10/microsoft-further-open-networking-specification?WT.mc_id=Blog_ServerCloud_Announce_CEA

P4, <http://yuba.stanford.edu/~nickm/papers/v44n3k2-bosshartA.pdf>

Broadcom OpenNSL, <http://blog.broadcom.com/network-infrastructure/inside-opennsl-open-source-innovation-in-the-network-has-arrived/>

(ONIE) Open Network Install Environment (<http://onie.org/>)

COMMON MANAGEMENT
TOOLS ACROSS NETWORK,
STORAGE AND COMPUTE
WORKING GROUP

2015



Open Networking
USER GROUP