Draft Architecture Principles

Architectural principles provide guidelines when making architecture decisions. The principles are *not* requirements (functional or non-functional), nor should they specify the design of the system.

1.Virtualized Software Based Applications

Flexibility in component deployment is important to ONAP functionality. All components in ONAP should be virtualized, preferably with support for both virtual machines and containers. All components should be software-based with no dependency on hardware platform. However, where hardware capabilities can enhance an ONAP component, those capabilities can be used if not required for the component to function.

2. Decoupled Applications Modules/Microservices

All components in ONAP should avoid tight coupling between other components. Components should be service-based with clear, concise function addressed by each service, following principles of microservices.

3. Shared Standard Cloud Platform

All components in ONAP should be able to run on a shared standards-based cloud platform. Ideally, the cloud platform implementation should be pluggable and transparent to the ONAP components.

4. Software Modularity and Reusability

Code within ONAP should be designed in a modular fashion & reusability is highly encouraged across components. Where applicable, reusable components can be provided as shared services across ONAP components.

5. Open Platform and APIs

An open, standards-based architecture with well-defined APIs fosters interoperability both within ONAP and across complementary projects and applications.

6. Supplier Agnostic

The ability for ONAP to be used by various users worldwide dictates the need to avoid dependency on any single supplier(s). All efforts within ONAP should be agnostic to a supplier's function, application, or code.

7. Model-Driven

All aspects of a network service design in ONAP should be model-driven, avoiding, where possible, programming code. This allows for a catalog-based reusable repository for a network service lifecycle.

8. Automation First

Embedding intelligence in the system rather than relying on expert resources fosters improved self-service with higher degrees of automation. When at all possible, actions within ONAP should be automated with minimal to no manual intervention required by operators.

9. Full Service Lifecycle Support

The ONAP platform should support the complete lifecycle of network services.

10. Agile Development and Operations (CI/CD)

ONAP is predicated on an accelerated lifecycle for network services. As such, agility is key in all aspects of ONAP: development of ONAP, designing of network services, and operation of both ONAP and network services. Principles of continuous integration and deployment should be followed by the ONAP platform.

11. Self Service

The ONAP platform should allow for users (such as vendors, product designers, operations, etc.) to perform functions within ONAP without the need to depend on code written by developers. Self-service is also enabled with user-focused design.

12. Availability

High availability and geographic redundancy are of key importance to many network services planned for ONAP. As such all ONAP components should be designed for high availability, resiliency, and geographic redundancy, including during upgrades.

13. Security

All ONAP components should keep security considerations at the fore-front of all architectural decisions. Security should be a pervasive underlying theme in all aspects of ONAP.

14. Pluggable Dependencies

When any ONAP component relies on software outside of the ONAP project, the dependency on that external software should be designed to pluggable, API-oriented, supporting multiple possible implementations of that dependency.

15. Integration Friendly

Many ONAP implementations will require integration with a variety of OSS/BSS systems. ONAP components should strive to provide clearly defined APIs for OSS/BSS integration, including standards-based interfaces where possible.

16. Policy-Based Decisions

ONAP components should strive to externalize any operational decisions into policies, allowing for dynamic definition of controls/behaviors and analytics.

17. Service, resource, function, and business model agnostic

Because of the variety of potential services and monetization models, ONAP component implementations should be careful to avoid assumptions about how a service might be used, the underlying resource or function, or the encompassing business model.

18. User-Focused

The needs of the users of ONAP (operators, developers, service designers, and more) should be central to the design of all user-facing components.

19. Common Data Layer and Information Model approach

ONAP components and platform should work with a uniform data layer which minimizes the number of data handling technologies. A consistent information model is desirable.

20. ONAP scaling

ONAP modules and platform should be scalable (both scaling up and scaling down). Modules should avoid using infrastructure resources when not providing work.

21. Multi-tenancy

The ONAP platform should support the ability manage multiple tenants and provide isolation for those tenants.

22. Centralized Design

All artifacts required for ONAP components should be able to be designed from a central design component.

23. Backward Compatibility

Backwards compatibility from release-to-release is essential to allow operators to take advantage of the features in a new release without the risk of breaking existing functionality. As such, all ONAP components should strive to ensure backward compatibility in their releases.