Controllers

A Controller manages the state of a single Resource (Application or Network). It executes the Resource's configuration and instantiation, and is the primary agent in ongoing management, such as control loop actions, migration, and scaling. All of these actions involve executing workflows obtained from Service Design and Creation (SDC). In addition, the Controller reports the status of each workflow execution to both the Active and Available Inventory (AAI) and the Master Service Orchestrator (MSO).

In effect, each Controller instance supports a form of orchestration to manage the operations that are within its scope. This orchestration is a lower-level of orchestration than the orchestration done by the MSO.

ONAP uses different Controller types to manage Resources in the execution environment, corresponding to their assigned controlled domain:

- Network configuration (**Network Controller**)
- Application (**Application Controller**).

Both these controllers are based on an OpenDaylight Controller Framework.

Network Controllers

A Network Controller, such as SDNC, instantiates a Virtual Network Function by carrying out its network configuration workflow and reporting the resulting status (to both AAI and MSO).

Examples of Network Controllers include those for Transport Virtual Network Functions, infrastructure networking (for instance, leaf, spine, and virtual switches), and Wide-Area-Networks (WANs).

<<DocRef: the SDN-C User Guide or SDNC API page, if there is one>>

The SDNC network controller in Figure 1 illustrates the core OpenDaylight framework that supports a model driven service abstraction layer (MD-SAL), api handlers, operational and configuration trees, and an adapter framework for integrating with controlled devices, virtual functions, and cloud infrastructure. Within this framework the ONAP Service Logic Interpreter (SLI) addition provides an extensible scripting language for expressing service logic through a Directed Graph (DG) builder based on Node-Red.   The SLI is extended by adding Java classes that can be called as a node in a DG to support frequent complex operations.

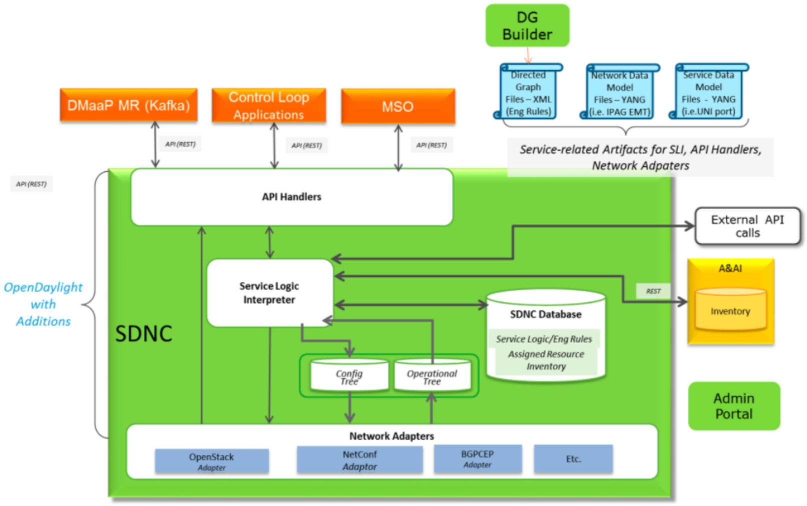
DG Builder
Directed Graph Files – XML (Eng Rules)
Network Data Model Files – YANG (i.e. IPAG EMT)
Service Data Model Files - YANG (i.e.UNI port)
Service-related Artifacts for SLI, API Handlers, Network Adpaters
DMaaP MR (Kafka)
Control Loop Applications
MSO
API (REST)
API (REST)
API (REST)
API (REST)
API Handlers
External API calls
A&AI
Inventory
REST
OpenDaylight with Additions
SDNC
Service Logic Interpreter
SDNC Database
Service Logic/Eng Rules
Assigned Resource Inventory
Config Tree
Operational Tree
Admin Portal
Network Adapters
OpenStack Adapter
NetConf Adaptor
BGPCEP Adapter
Etc.

Figure 1. SDNC Network Controller

<<TODO Find Reference and example DG and SLI extension node in a Gerrit Project>>

Application Controllers

Application controllers, such as APPC, receive orchestrated requests from the MSO, which obtains application-specific components and attributes from SDC. The MSO continues to be responsible for ensuring that the Application Controller successfully completes its Resource configuration as defined by the workflow.

Note that not all changes in network or service behavior are the result of orchestration. Policies and rules (in the Policy subsystem) inform the controller such that the Controller can enable service behavior changes.

ONAP includes a generic Application Controller (APPC, formerly called APP-C) that receives commands from ONAP components, such as MSO, DCAE, or the Portal, and uses these commands to manage the life cycle of Services, Resources (virtual applications and Virtual Network Functions), and their components. The Application Controller is based on the same OpenDaylight Framework with SLI additions.   The Application Controller project provides SLI node types, adapters, and API handling logic required for the life cycle management of complex, distributed services composed of many VNF types and instances.

Documentation:

- ONAP Application Controller User Guide
- APPC API Documentation
- APP-C Ansible Adapter Documentation
- APP-C Chef Adapter Documentation

NOTE: For instructions on how to build and run an ONAP Component (APP-C is taken as an example) locally, please refer to **this documentation.**