ARC Controller Component Description – Istanbul-R9

Page Status: Updated for Istanbul - 22 Mar 2021

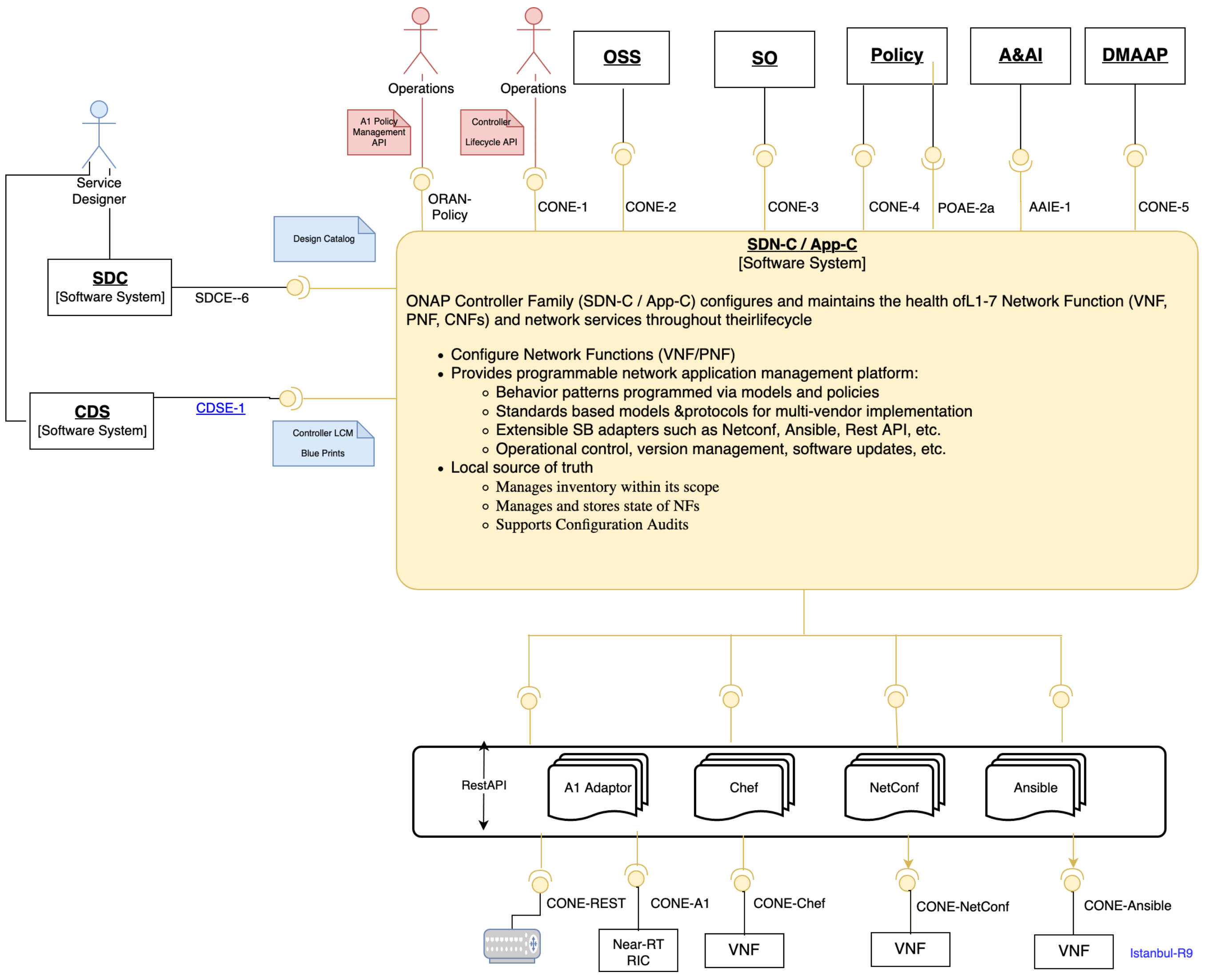
Component Status: Reviewed 29 Jun 2021

Last Reviewed on: 29 Jun 2021

Certified by: Dan Timoney

1. High Level Component Definition and Architectural Relationships

SDN-C / App-C:



ONAP Controller Family (SDN-C / App-C) configures and maintains the health ofL1-7 Network Function (VNF, PNF, CNFs) and network services throughout theirlifecycle

- Configure Network Functions (VNF/PNF)
- Provides programmable network application management platform:
  - Behavior patterns programmed via models and policies
  - Standards based models &protocols for multi-vendor implementation
  - Extensible SB adapters such as Netconf, Ansible, Rest API, etc.
  - Operational control, version management, software updates, etc.
- Local source of truth
  - Manages inventory within its scope
  - Manages and stores state of NFs
  - Supports Configuration Audits

Note: ONAP has two application level configuration and lifecycle management modules called SDN-C and App-C. Both provide similar services (application level configuration using NetConf, Chef, Ansible, RestConf, etc.) and life cycle management functions (e.g. Stop, resume, health check, etc.). They share common code from CCSDK repo. However, there are some differences between these two modules (SDN-C uses CDS only for onboarding and configuration / LCM flow design, whereas App-C uses CDT for the LCM functions for self service to provide artifacts storing in APP-C Database). SDN-C has been used mainly for Layer1-3 network elements and App-C is being used for Layer4-7 network functions. This is a very loose distinction and we expect that over time we will get better alignment and have common repository for controller code supporting application level configuration and lifecycle management of all network elements (physical or virtual, layer 1-7). Because of these overlaps, we have documented SDN-C and App-C together.

ONAP Controller Family (SDN-C / App-C) configures and maintains the health ofL1-7 Network Function (VNF, PNF, CNFs) and network services throughout their lifecycle

- Configure Network Functions (VNF/PNF)
- Provides programmable network application management platform:
  - Behavior patterns programmed via models and policies
  - Standards based models &protocols for multi-vendor implementation
  - Extensible SB adapters such as Netconf, Ansible, Rest API, etc.
  - Operational control, version management, software updates, etc.
- Local source of truth
  - Manages inventory within its scope
  - Manages and stores state of NFs
  - Supports Configuration Audits

2. SDN-C/APP-C API definitions

Controller provides the following interfaces:

Interface Name	Interface Definition	Interface Capabilities	API Spec (Swagger)
ORAN-Policy	A1 policy management updates	An interface to accept policy management updates for distribution to managed non-RT RICs	A1 Policy Management API (yaml)
CONE-1	Operations Interface APP-C : LCM	An interface to request for Lifecycle management operations on network resources. This is the same interface as CONE-2, but is invoked by a command line tool (e.g. curl) instead of by a system.	No Swagger, but documented in ReadTheDocs
CONE-2	OSS Interface APP-C : LCM	An interface to request for Lifecycle management operations on network resources	No Swagger, but documented in ReadTheDocs
CONE-3	Service Order Interface (GENERIC-RESOURCE-API)	An interface to request for Configuration and Lifecycle management operations on network resources	GENERIC-RESOURCE-API swagger (yaml)
CONE-4	Policy Interface SDN-C: LCM	An interface to support LCM requests such as Restart, Rebuild, Migrate, Evacuate operations on network resources (APP-C interfaces with openstack to send those LCM requests to VNF/VNF-C/VM)	Swagger TBD - Interface format is the same as APP-C LCM (see ReadTheDocs)

The current API documents can be found at:

https://onap.readthedocs.io/en/casablanca/submodules/appc.git/docs/index.html

Controller consumes the following interfaces:

Interface Name	Interface Definition	Interface Capabilities	API Spec (Swagger)
CONE-5	Rest API	An interface for communication with external systems such as IP management	
CONE-6	Resource Chef API	An interface for configuration and Lifecycle management of network resources using Chef protocol	
CONE-7	Resource NetConf API	An interface for configuration and Lifecycle management of network resources using NetConf protocol	
CONE-8	Resource Ansible API	An interface for configuration and Lifecycle management of network resources using Ansible protocol	
SDCE-6	SDC Interface	An interface to receive resource Templates from SDC design catalog	
CDSE-1	CDS Interface	An interface to receive resource blueprint from CDS	
AAIE-1	Inventory Service Interface	An interface to create, update, query, and delete resource information and relationships	

POE-2a	PDP Query API	Policy Decision Point query for IP address	

3. Component Description:

blocked URL

4. known system limitations

- Lack of clarity & roles in the controller (which controller does what?)
- Proliferation of controller instances (many similar yet different controller instances)
- Divergence of controller implementation (lack of common controller framework)
- Duplicate and uncoordinated interfaces (lack of uniform common services in southbound interfaces)
- Controller resiliency and horizontal scaling

5. Used Models

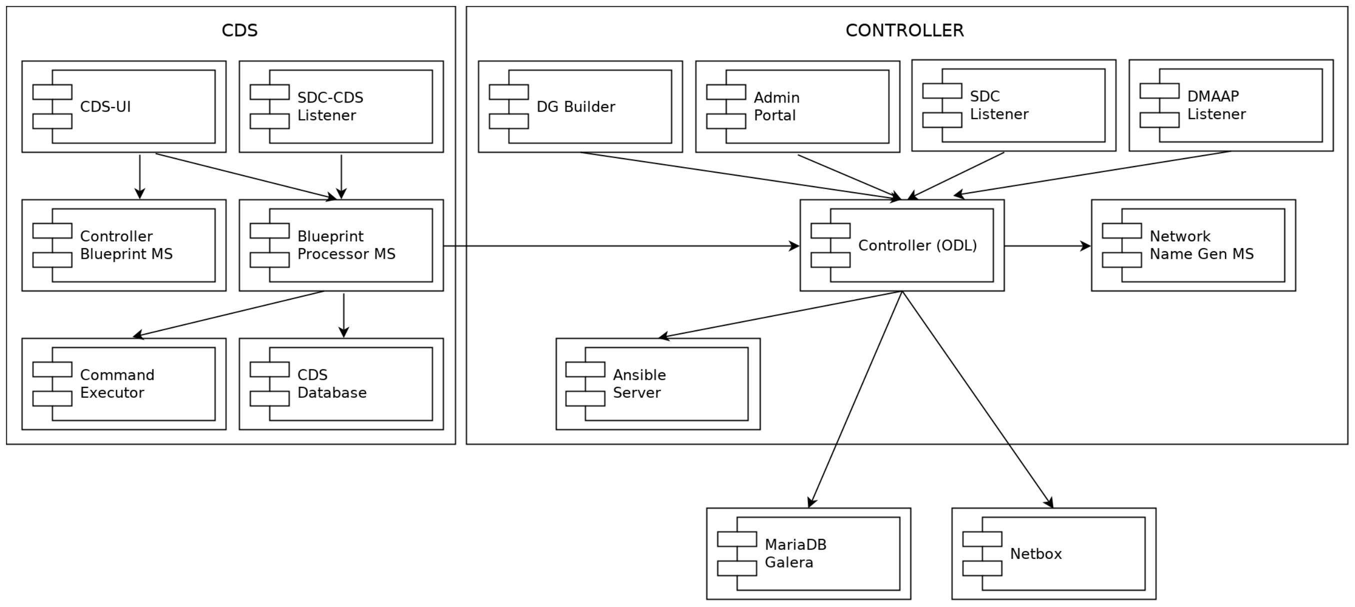
Controllers use the following models:

- TOSCA
- YANG

6. System Deployment Architecture

Controller consists of the following containers:

- CDS-UI container - Design Time
- SDC-CDS Listener container - Design Time
- CDT-UI container - Design Time
- Controller Blueprint MS Container - Design Time
- Blueprint Processor MS Container - Run Time
- Command Executor Container - Run Time
- CDS Database Container - Run Time
- DG Builder Container - Design Time
- Admin Portal Container - Run Time
- SDC Listener Container - Run Time
- DMAAP Listener Container - Run Time
- Controller (ODL) Container - Run Time
- Network Name Gen MS Container - Run Time
- Ansible Server Container - Run Time
- MariaDB Galera - Platform Service
- Netbox - Platform Service
- RunTime Config DB - Run Time
- A1 Policy Manager - Run Time



7. New Release Capabilities

- Upgrade of ODL to Silicon SR (Service Release) 1
- OpenDaylight separation:
    - For Istanbul, we plan to provide 2 implementations of GENERIC-RESOURCE-API:
        - The current implementation, implemented as an OSGi feature deployed in the OpenDaylight karaf container.
        - A new springboot-based microservice, which runs in a pod outside of OpenDaylight

    We eventually would like to replace the current implementation with the springboot-based implementation, and run OpenDaylight as a separate pod. However, there is still some work required to close feature gaps in the springboot-based implementation before we can do so without breaking existing functionality.

    In Istanbul, we plan to conduct a proof of concept of the work done to date. For this proof of concept, we will replace the SDNC controller in our local kubernetes test environment with the springboot-based version and run the standard OOM gating test suite to determine whether that the implementation to date is fully backwards compatible. Any issues identified will be tracked in Jira so that we can plan to close any gaps discovered so that we can plan to introduce our springboot-based platform as the default SDNC implementation in Jakarta.

8. References

1. APP-C overview & User Guide: https://docs.onap.org/en/casablanca/guides/onap-developer/developing/index.html#application-controller
2. SDN-C overview & User Guide: https://docs.onap.org/en/casablanca/guides/onap-developer/developing/index.html#software-defined-network-controller