

AAI R8 Honolulu Release Planning

The content of this template is expected to be fill out for M1 Release Planning Milestone.

- 1 [Overview](#)
- 2 [Scope](#)
 - 2.1 [What is this release trying to address?](#)
 - 2.2 [Requirements](#)
 - 2.3 [Minimum Viable Product](#)
 - 2.3.1 [Epics](#)
 - 2.3.2 [Stories](#)
 - 2.4 [Longer term roadmap](#)
- 3 [Release Deliverables](#)
- 4 [Sub-Components](#)
- 5 [Architecture](#)
 - 5.1 [High level architecture diagram](#)
 - 5.2 [Platform Maturity](#)
 - 5.3 [API Incoming Dependencies](#)
 - 5.4 [API Outgoing Dependencies](#)
 - 5.5 [Third Party Products Dependencies](#)
- 6 [Testing and Integration Plans](#)
- 7 [Gaps](#)
- 8 [Known Defects and Issues](#)
- 9 [Risks](#)
- 10 [Resources](#)
- 11 [Release Milestone](#)
- 12 [Team Internal Milestone](#)
- 13 [Documentation, Training](#)
- 14 [Other Information](#)
 - 14.1 [Vendor Neutral](#)
 - 14.2 [Free and Open Source Software](#)

Overview

Project Name	Enter the name of the project
Target Release Name	Honolulu
Project Lifecycle State	Mature. Refer to ONAP Charter, section 3.3 Project Lifecycle for further information
Participating Company	AT&T, Amdocs, Bell Canada, China Mobile, Huawei, Orange, Wipro, YoppWorks

Scope

What is this release trying to address?

The scope of the Honolulu Release of AAI will address Global Requirements, some functionality updates, as well as the approved use cases.

Highest priority will be assigned to support global requirements. As next highest priority, AAI will support the approved use cases. Best-effort additional development will continue to support additional features not included in the approved use cases.

[AAI R8 Honolulu Release: Global Requirements](#)

Requirements

Use Cases: [AAI R8 Honolulu Release: Functional Use Cases](#)

TSC MUST HAVES : [AAI R8 Honolulu Release: Global Requirements](#)

Minimum Viable Product

- A&AI Core: Resources - CRUD REST API endpoints for A&AI inventory resources
- A&AI Core: Traversal - REST APIs for graph traversals. Some APIs use the Gremlin Server
- A&AI Core: Babel - Parses models received from SDC

- A&AI Core: Event-Client - publishes an event stream of changes to the database
- A&AI Core: Model-loader - Loading models parsed from SDC
- A&AI Core: Logging-service - Service to standardize logging for compliance and readability
- A&AI Core: GraphAdmin - suite of functionality to manage the graph database
- UI - An ONAP portal application used by operations for visualizing and searching
- Rest-Client - A library for an user to interface with the A&AI core APIs of resources and traversal

Functionalities

List the functionalities that this release is committing to deliver by providing a link to JIRA Epics and Stories. In the JIRA Priority field, specify the priority (either High, Medium, Low). The priority will be used in case de-scoping is required. Don't assign High priority to all functionalities.

Epics

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
-----	---------	---	---------	---------	-----	----------	----------	---	--------	------------

No issues found

Stories

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
-----	---------	---	---------	---------	-----	----------	----------	---	--------	------------

No issues found

Longer term roadmap

- In future release, AAI looks toward additional simplification of visualization of the data model
- Increased S3P, focusing on security and reducing turn around on 3rd party dependency vulnerabilities
- Historical Data Tracking down to the attribute-level
- Enhanced UI with animated state topology
- AAI looks toward geo-redundant clusters of the Cassandra database
- Tabular data views / aggregate views

Release Deliverables

Indicate the outcome (Executable, Source Code, Library, API description, Tool, Documentation, Release Note, etc) of this release.

Deliverable Name	Deliverable Description
A&AI resources	Executable. REST CRUD interface to graph database
A&AI traversal	Executable. REST Interface for complex queries
Router-core	Library
Sparky-fe	Library
Sparky-be	Executable (AAI UI)
Schema-service	Executable (delivers schema to other AAI mS)
Graphadmin	Executable (graph administration functions)

Sub-Components

Please see the INFO.yaml files associated with each repo as the authoritative sources of information. <https://gerrit.onap.org/r/admin/repos/q/filter:aai>

Architecture

High level architecture diagram

At that stage within the Release, the team is expected to provide more Architecture details describing how the functional modules are interacting.

Indicate where your project fit within the [ONAP Architecture diagram](#).

Block and sequence diagrams showing relation within the project as well as relation with external components are expected.

Anyone reading this section should have a good understanding of all the interacting modules.

Platform Maturity

Please fill out the centralized wiki page: [Honolulu Release Platform Maturity](#)

AAI data is filled out

• API Incoming Dependencies

List the API this project is expecting from other projects.

Prior to Release Planning review, Team Leads must agreed on the date by which the API will be fully defined. The API Delivery date must not be later than the release API freeze date.

Prior to the delivery date, it is a good practice to organize an API review with the API consumers.

API Name	API Description	API Definition Date	API Delivery date	API Definition link (i.e.swagger)
DMaaP	Eventing bus			
MSB	Service registry			
SDC	AAI Model loader consumes model objects			
AAF	Authentication / Authorization			

API Outgoing Dependencies

API this project is delivering to other projects.

API Name	API Description	API Definition Date	API Delivery date	API Definition link (i.e.swagger)

• Third Party Products Dependencies

Third Party Products mean products that are mandatory to provide services for your components. Development of new functionality in third party product may or not be expected.

List the Third Party Products (OpenStack, ODL, RabbitMQ, Elasticsearch, Crystal Reports, ...).

Name	Description	Version
JanusGraph	Open-source, distributed graph database	0.2.3
Cassandra	Open-source distributed storage system	3.11
Docker	VM container	
SpringBoot	Open-source Microservice application sever	1.5.22 or 2.x
HAPROXY	microservice front end	1.8
Centos/Alpine	OS	See docker config files

• Testing and Integration Plans

Provide a description of the testing activities (unit test, functional test, automation,...) that will be performed by the team within the scope of this release.

[AAI R8 Honolulu: Functional Test Cases](#)

• Gaps

This section is used to document a limitation on a functionality or platform support. We are currently aware of this limitation and it will be delivered in a future Release.

List identified release gaps (if any), and its impact.

Gaps identified	Impact
To fill out	To fill out

• Known Defects and Issues

TBD

• Risks

List the risks identified for this release along with the plan to prevent the risk to occur (mitigation) and the plan of action in the case the risk would materialized (contingency).

Please update any risk on the centralized wiki page - [Honolulu Risks](#)

• Resources

AT&T, Amdocs, Bell Canada, China Mobile, Huawei, Orange, Wipro, YoppWorks

• Release Milestone

The milestones are defined at the [Release Planning: Honolulu](#) and all the supporting project agreed to comply with these dates.

• Team Internal Milestone

This section is optional and may be used to document internal milestones within a project team or multiple project teams. For instance, in the case the team has made agreement with other team to deliver some artifacts on a certain date that are not in the release milestone, it is recommended to provide these agreements and dates in this section.

It is not expected to have a detailed project plan.

Date	Project	Deliverable
To fill out	To fill out	To fill out

• Documentation, Training

Please update the following centralized wiki: [Honolulu Documentation](#)

That includes

- Team contributions to the specific document related to the project (Config guide, installation guide...).
- Team contributions to the overall Release Documentation and training asset
- High level list of documentation, training and tutorials necessary to understand the release capabilities, configuration and operation.
- Documentation includes items such as:
 - Installation instructions
 - Configuration instructions
 - Developer guide
 - End User guide
 - Admin guide
 - ...

Other Information

• Vendor Neutral

If this project is coming from an existing proprietary codebase, ensure that all proprietary trademarks, logos, product names, etc. have been removed. All ONAP deliverables must comply with this rule and be agnostic of any proprietary symbols.

• Free and Open Source Software

FOSS activities are critical to the delivery of the whole ONAP initiative. The information may not be fully available at Release Planning, however to avoid late refactoring, it is critical to accomplish this task as early as possible.
List all third party Free and Open Source Software used within the release and provide License type (BSD, MIT, Apache, GNU GPL,...).
In the case non Apache License are found inform immediately the TSC and the Release Manager and document your reasoning on why you believe we can use a non Apache version 2 license.

Each project must edit its project table available at [Project FOSS](#).

Charter Compliance

The project team comply with the [ONAP Charter](#).