


Holmes Release Planning for Kohn

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

 **Info**

Use the "Copy" and "Move" options (available under the ..., top right of this page) to duplicate this template into your project wiki.

Use the Wiki to document the release plan. Don't provide PowerPoint.

Use as many diagrams and flow charts as you need, directly in the wiki, to convey your message.

- 1 [Overview](#)
- 2 [Scope](#)
 - 2.1 [What is this release trying to address?](#)
 - 2.2 [Requirements](#)
 - 2.3 [Minimum Viable Product](#)
 - 2.4 [Functionalities](#)
 - 2.4.1 [Epics](#)
 - 2.4.2 [Stories](#)
 - 2.5 [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	Kohn
Project Lifecycle State	Incubation
Participating Company	ZTE, CMCC, HUAWEI, Fujitsu

Scope

What is this release trying to address?

- bugfix
- platform maturity improvements

Requirements

None.

Minimum Viable Product

- rules (changes may be introduced by bugfix) for CCVPN/VoLTE/MDONS

- the rule management component
- the engine management component

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
HOLMES-588	Global Requirements Approval		Nov 03, 2022	Nov 03, 2022	Jun 09, 2022	Unassigned	None	==	CLOSED	Won't Do
HOLMES-584	Release Sign Off		Oct 27, 2022	Nov 10, 2022	Nov 10, 2022	Unassigned	None	==	CLOSED	Done
HOLMES-575	Release Candidate		Oct 03, 2022	Oct 27, 2022	Oct 20, 2022	Unassigned	None	==	CLOSED	Done
HOLMES-566	Feature Freeze		Sep 02, 2022	Sep 26, 2022	Sep 15, 2022	Unassigned	None	==	CLOSED	Done
HOLMES-557	Finalized Code Submission		Jul 19, 2022	Nov 10, 2022	Sep 01, 2022	Unassigned	None	==	CLOSED	Done
HOLMES-547	Specification Freeze		Jun 27, 2022	Nov 10, 2022	Jun 30, 2022	Unassigned	None	==	CLOSED	Done
HOLMES-537	Global Requirements Approval		May 09, 2022	Nov 10, 2022	May 26, 2022	Unassigned	None	==	CLOSED	Done

7 issues

Stories

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
HOLMES-511	Migrate from Dropwizard to Springboot		Dec 28, 2021	Sep 16, 2022		Unassigned	None	==	CLOSED	Done

1 issue

Longer term roadmap

- integration with Acumos
- integration with ORAN

Release Deliverables

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

Deliverable Name	Deliverable Description
API description	A brief introduction of the APIs of Holmes. Both external and internal users (systems) could implement alarm analyses using these APIs
Documentation	Installation manual, user guide, etc. Please refer to Kohn Documentation .
Release Notes	Release notes of the release

Source Code	The source code of the sub-components is listed below.
-------------	--

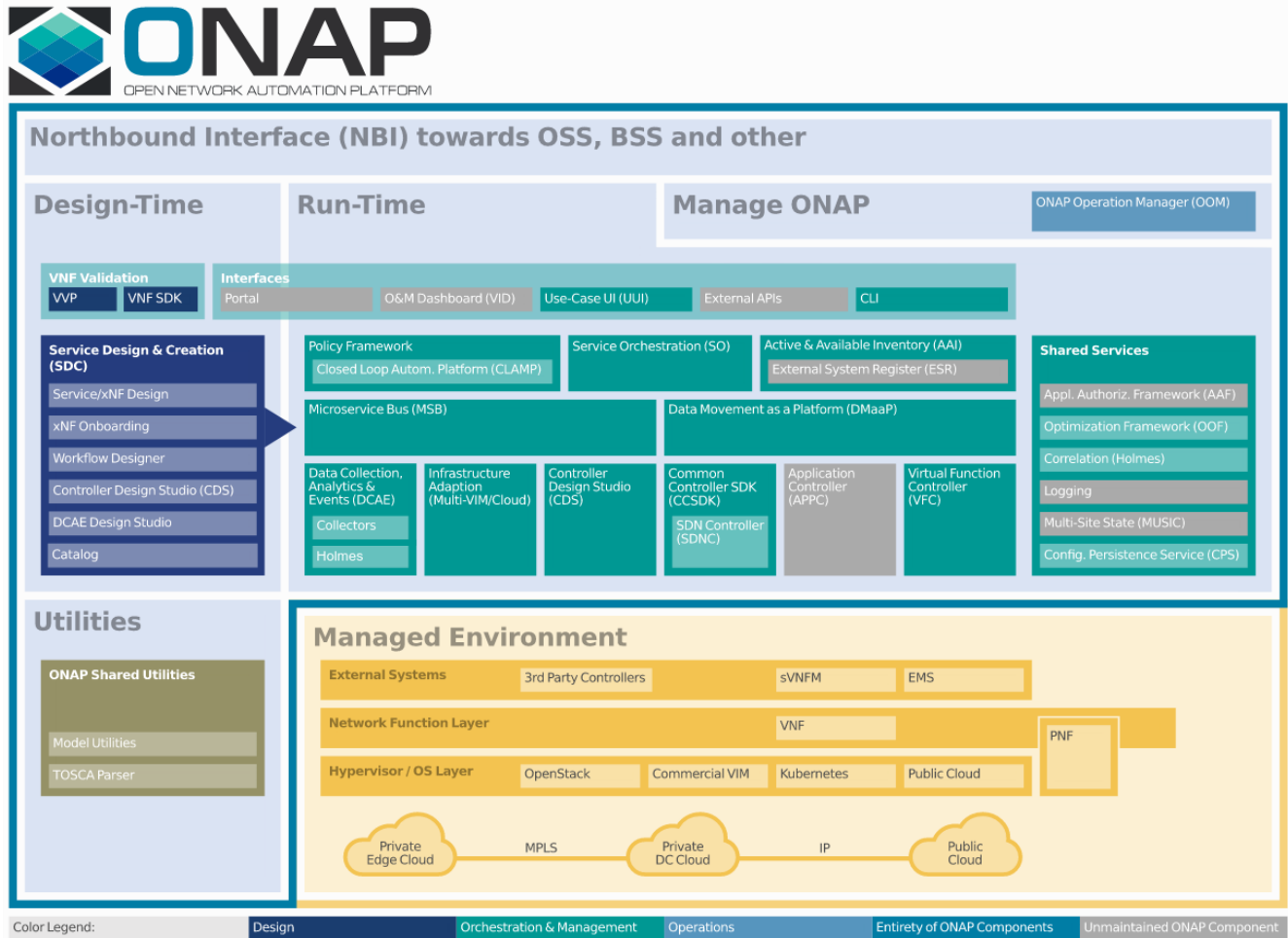
Sub-Components

- the Rule Management component
- the Engine Management component

Architecture

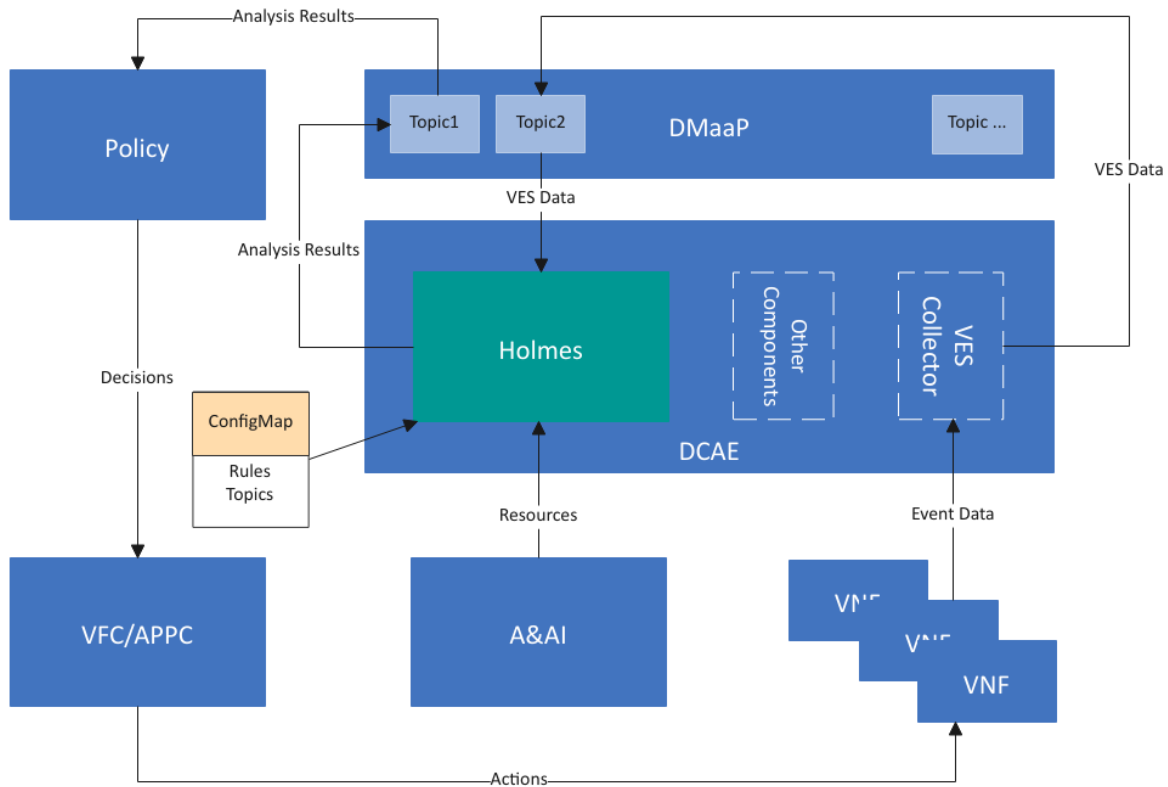
High level architecture diagram

Holmes is architecturally an analytics application that resides within DCAE.

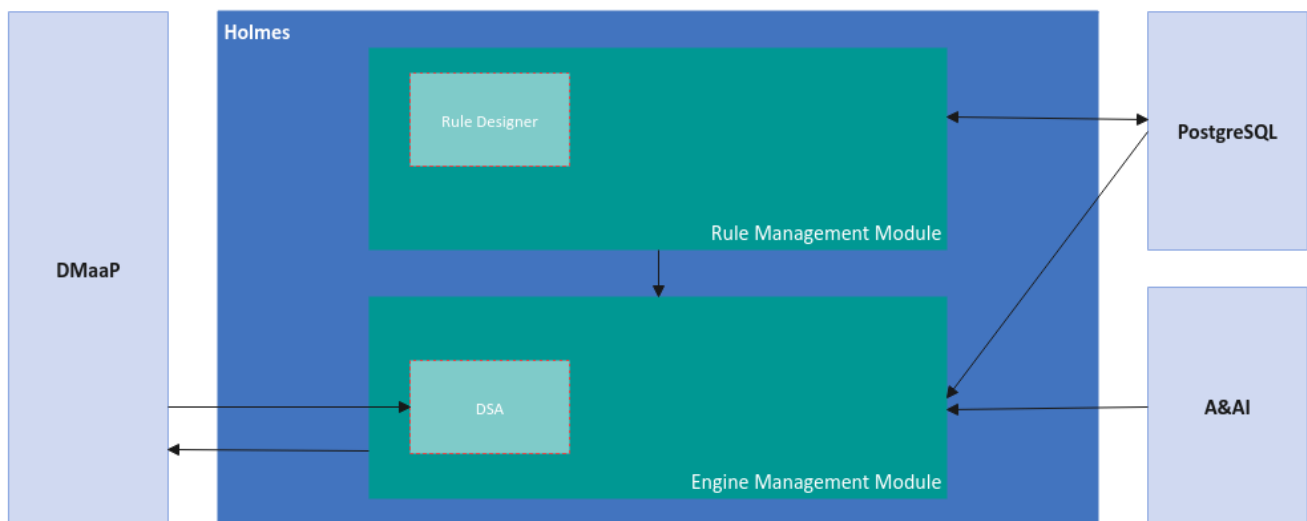


Normally, it is deployed by DCAE. But if users want to use Holmes independently (without DCAE), it could also be deployed in a standalone mode in the form of ordinary docker containers.

The interaction diagram between Holmes and its relative components is depicted below:



Holmes itself consists of two basic sub-modules: the rule management module and the engine management module. The rule management module is mainly responsible for the CRUD operations of Holmes rules and persisting the rules into a database. The engine management module uses the Drools engine as its core component to support correlation analysis among alarms. The module diagram is like below:



Platform Maturity

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

API Incoming Dependencies

List the API this project is expecting from other projects.

Prior to the Release Planning review, Team Leads must agree 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)
Data Movement as a Platform APIs	DMaaP message sub/pub related APIs which will be used by Holmes to collect the data from and publish data to DMaaP topics.			Data Movement as a Platform Message Router DMaaP Message Router API
Resource Query	Query different resource information from A&AI. All A&AI operations are implemented in the form of RESTful APIs. I'm using "Resource Query" as a general name for the APIs in case there will be too many APIs listed here.			AAI API
Service Registration /Un-registration Service Discovery	The APIs used to register/un-register a micro-service to/from MSB . The APIs used to discover another micro-service via MSB.			Microservice Bus API Documentation

API Outgoing Dependencies

There's no API intended for other projects. The internal APIs are like following (there have been no modifications on APIs since Dublin, so we'll reuse the docs from Dublin.):

API Name	API Description	API Definition Date	API Delivery date	API Definition link (i.e. swagger)
Rule Creating	This API is intended for creating a rule in the database.	28 Jun 2017	August, 24th, 2017	Rule Management - Dublin
Rule Modifying	This API is intended for modifying a rule in the database.	28 Jun 2017	August, 24th, 2017	Rule Management - Dublin
Rule Deleting	This API is intended for deleting a rule from the database.	28 Jun 2017	August, 24th, 2017	Rule Management - Dublin
Rule Query	This API is intended for querying rules from the database.	28 Jun 2017	August, 24th, 2017	Rule Management - Dublin
Health Check	This API is used by other components to check whether Holmes is working.	17 Aug 2017	August, 24th, 2017	Health Check - Dublin

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
Drools (JBoss Rules)	Drools is a Business Rules Management System (BRMS) solution. It provides a core Business Rules Engine (BRE), a web authoring and rules management application (Drools Workbench) and an Eclipse IDE plugin for core development.	7.62.0. Final
PostgreSQL	PostgreSQL is used for the sake of data (holmes rules) persistence.	9.5.0

In case there are specific dependencies (Centos 7 vs Ubuntu 16. Etc.) list them as well.

Testing and Integration Plans

- For unit tests, we are going to keep the line coverage of each module to be above 55% or even higher.

- For the functional tests, because there will be few functional requirements for Holmes, we will main reuse the current auto-testing scripts to promise that the basic functions of Holmes work well. As for the GUI part, we will add some new test cases to the wiki page and attach the corresponding testing report to it.
- For the non-functional requirements, we will set up a set of testing env and get them tested by ourselves. Meanwhile, we'll be collaborating with the integration team to get some advice on how to get all those platform maturity requirements tested.

Gaps

This section is used to document a limitation on 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 their impact.

Gaps identified	Impact
To fill out	To fill out

Known Defects and Issues

None.

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 - [Kohn Risks](#)

Resources

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

Release Milestone

The milestones are defined at the [Release Planning: Kohn](#) and all the supporting projects 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 teams to deliver some artifacts on a certain date that are not in the release milestone, it is erecommended 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: [Kohn 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
 - ...



Note

The Documentation project will provide the Documentation Tool Chain to edit, configure, store and publish all Documentation asset.

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](#).