
# ONAP Policy Framework

## Frankfurt

## Building Policy Enforcement into your application and using the Decision API

Pamela Dragosh – PTL

July 2020

# What applications is this appropriate for?

- Any ONAP component or application that wants to be policy-enabled and enforce policies during runtime
- Current ONAP components that are policy-enabled:
  - DCAE analytics/collectors – support Monitoring Policy Types
  - OOF – support Optimization Policy Types
  - SDNC – support Naming Policy Types

# Get The Policy Components Running Locally

- Docker Compose script
  - There is a docker-compose.yml script saved in the source code tar file:
    - src/main/docker/docker-compose.yml
  - To start it run these commands:
    - docker-compose -f docker-compose.yml run --rm start_dependencies
    - docker-compose -f docker-compose.yml run --rm start_all
  - NOTE: You may have to tweak the script to get it running in your environment.

# Test That The Policy Components Are Running Locally

- POSTMAN Collection
  - If you have POSTMAN application, there is a collection saved in the tar that you can use to test that the components are successfully running in your local environment.
  - src/test/resources/PolicyEnforcementTutorial.json

# Policy Type Design

- Design the Policy Type first and what you expect the policies to look like
  - There are many examples already available in ONAP to reference
  - https://github.com/onap/policy-models/tree/master/models-examples/src/main/resources
  - This tutorial will use a very simple Policy Type that inherits from Monitoring base policy type since these types of dynamic policy types are supported “out-of-the-box” in ONAP

# Load the Policy Type and Policies into the Policy Framework Platform components

- Once you have designed a Policy Type, you can use the Policy Lifecycle API to CRUD the Policy Type and its Policies
- This tutorial contains a Postman collection and the README.txt contains curl command examples that can be used to:
  - Create Policy Type
  - Create a Policy for the Policy Type

# Deploy the Policy to the XACML PDP Engine

- This tutorial will be specific to applications that need to use the Decision API (implemented by the XACML PDP) to support policy enforcement.
  - Note: for apex or drools supported Policy Types, please see the tutorial for each to understand how to build enforcement applications for these PDPs.
- Deploy the policy

# What code do I need to develop in my application?

- Your application should be able to make a RESTful API call
- Your application should be able to parse the JSON payload of the RESTful API call
- If your application needs notification of a policy change, it should be able to connect to Dmaap to receive Policy Update notifications
- Is there an SDK in Policy to support this?
  - Not exactly an SDK, but Java-based applications can use our java artifacts in our policy/common and policy/models repositories if they do not have code in their codebase that can perform RESTful API calls

# Demo – getting started

- Eclipse can be used to build a new Java Maven application
- Create an empty maven Java project
  - Ensure using JDK 11 for compilation as the Policy Framework only supports JDK 11
- Find the latest ONAP Policy Frameworks java artifacts and Docker images by consulting our wiki page:
  - https://wiki.onap.org/display/DW/Policy+Framework+Project%3A+Component+Versions
  - This tutorial will use policy/common and policy/models java artifacts

# Tutorial

- Tutorial – the source code created will be uploaded to the wiki:
  - https://wiki.onap.org/pages/viewpage.action?pageId=84654890

# For more information

- Developer Documentation is located here:
  - https://docs.onap.org/projects/onap-policy-parent/en/frankfurt/development/development.html