ONAP
OPEN NETWORK AUTOMATION PLATFORM

ONAP Policy Framework

Frankfurt
XACML PDP Application Design and understanding how the Decision API is implemented
Pamela Dragosh – PTL

August 2020

Who is this tutorial for?

- ONAP Developers that wish to use Decision API in their custom application to enforce policies during runtime.
  - Eg. Custom PEP (Policy Enforcement Point)
  - Note: Separate tutorial to build PEP into your application
    - https://wiki.onap.org/pages/viewpage.action?pageId=84654890

Who is this tutorial for? (continued)

- The current set of applications in the ONAP XACML PDP do not meet your needs
  - Each application translates policies and responds to the Decision API appropriately. If your PEP has different needs, then you can build a custom XACML PDP Application.
    - Wish to add another layer of algorithm’s on top of XACML PDP engine decision.
      - Eg. Extend the existing translators or override some of their methods and change the behavior
    - Prefer to translate TOSCA Policies differently
      - Eg. Write your own translator
    - Desire to segregate “actions” into your own application
      - An “action” is used in the Decision API payload to forward a Decision request to an application.
      - Applications can have more than one action associated with their application

Where to find what applications/translators are currently available

- XACML PDP Frankfurt Documentation
  - https://docs.onap.org/projects/onap-policy-parent/en/frankfurt/xacml/xacml.html
- Clone the XACML PDP codebase (frankfurt branch)
  - https://github.com/onap/policy-xacml-pdp/tree/frankfurt
  - Study the “applications” sub-module to see how the current set of applications are built.
    - Monitoring and Naming
    - Guard
    - Optimization
    - Native

XacmlApplicationServiceProvider interface

- XacmlApplicationServiceProvider interface
  - This is the interface that an application is required to implement
  - The XACML PDP uses java.service to find implementations of this service interface to load into the PDP and make available
  - **StdXacmlApplicationServiceProvider** is an implementation of this interface that performs a lot of common work that the packaged applications utilize.
    - Strongly recommend using this interface and overriding methods as appropriate

ToscaPolicyTranslator interface

- ToscaPolicyTranslator interface
  - Must be provided to the implementation of the XacmlApplicationServiceProvider for policy translation
  - There are some implementations available for use:
    - StdCombinedPolicyResultsTranslator – very basic, not recommended
    - StdMatchableTranslator – recommended, allows flexibility of attributes to be used in policy design
  - Possible to create your own translator if desired

Building the application

- Watch the video!!
  - Code is uploaded into this wiki:
  - https://wiki.onap.org/pages/viewpage.action?pageId=84654893
- Tutorial is documented in readthedocs and the code is available there as well
  - https://docs.onap.org/projects/onap-policy-parent/en/frankfurt/xacml/xacml-tutorial.html

Testing against running ONAP Policy Framework components

- Clone the policy/models codebase (frankfurt branch)
  - https://github.com/onap/policy-models/tree/frankfurt
  - This repo contains a dmaap-simulator that we will be using to test our new application
  - Build the dmaap-simulator
    - mvn clean install
    - cd models-sim/models-sim-dmaap
    - bash ./src/main/package/docker/docker_build.sh
  - Running ‘docker images’ should show the following:

```
$ docker images
REPOSITORY                                  TAG                             IMAGE
ID            CREATED            SIZE
dmaap/simulator                             latest                          24364765dd49        6
seconds ago        414MB
```

POSTMAN Collection

- There is a POSTMAN collection available to test the application
  - Validate all the healthchecks work for API, PAP and Tutorial version of the XACML-PDP
  - Create the new Authorization Policy Type and Policies via the Lifecycle API
  - Because we are using out-of-box images, we need to Delete the default group and re-build it to include our new Policy Type
    - NOTE: It is assumed that one would override the default configurations for API and PAP components with your Authorization Policy Type already loaded and the default PDP group with the custom tutorial app advertising that Policy Type as supported in a production environment.
  - Deploy and test the Decision API
    - You can also test UnDeploy of policies to see how the Decision response would change