Demo: Adaptive Policy Execution (APEX)

Sven van der Meer, John Keeney, Liam Fallon, and Joseph McNamara
Network Management Lab, Ericsson, Athlone, Co. Westmeath, Ireland
{sven.van.der.meer, john.keeney, liam.fallon, joseph.mcnamara}@ericsson.com

Abstract—This demo focuses on demonstrating features of the Adaptive Policy Execution (APEX) system. APEX is a carrier-grade, production ready, scalable policy engine implementing, based on published theory, universal and immutable policy infrastructure. The demo will showcase the main APEX features, from authoring via deployment to runtime; with three demo use cases. All software and features are available on Github.

I. INTRODUCTION

The Adaptive Policy Execution (APEX) is a carrier-grade, production ready, scalable policy authoring, deployment, and execution environment. It is based on our Universal Policy Theory (UPT) implementing immutable policy infrastructure according to the Universal Policy Execution Environment (UPEE); both detailed in [1]. We have discussed the requirements for adaptive policies in [2] and a SON use case in [3]. The original design and implementation of APEX was introduced in [4]. A use case for an adaptive management of mobile network security is described in [5].

APEX also features a novel implementation for context-aware policies with distributed context detailed in [6]. 5G related aspects of policy evolution and requirements, including closed control loop management, a novel policy model, and the underlying resource modelling have been discussed in [7].

The APEX software was released as open source by Ericsson in January 2018. The source code is available at Github [8] and the documentation is available at [9]. APEX has been submitted to the ONAP [10] Policy Framework project for adoption as a new Policy Decision Point (PDP) type.

II. THE APEX SYSTEM

The APEX (Adaptive Policy Execution) policy engine is a fully featured policy engine that executes anything from simple to adaptive policies. An adaptive policy is one that can modify its behavior based on the current conditions of the network and systems. Adaptive policies have the following characteristics:

- Are able to make decisions at runtime rather than simply selecting decisions described at policy definition time
- Can use additional context information that was not provided in the incoming event or request
- Can self-adapt at runtime to modify decision making

The APEX policy engine accepts input events and requests from other components, routes the input to the appropriate policies, computes the policy results, and generates response events or actions to be processed by other components. The policies may be affected by information injected into the policy context as changes in business or domain goals, by information derived from previous executions of the policies, and by context information retrieved from other components (analytics, inventory, topology, etc.).

APEX connects to a Trigger System to receive events that trigger policies and to an Actioning System that receives policy results. APEX supports various connection technologies, such as messaging systems (Kafka, Websockets), REST, and file input/output. APEX supports context (as additional information) for all events as well as inside the engine to support policies. Context information can be read from any outside source.

A policy is defined in a Universal Execution Policy Specification (UEPS), directly executable in an APEX engine. Higher-level policy specifications (or existing policy languages) can be easily translated in UEPS.

An APEX system can use multiple policy engines with different policies deployed on each of them. Context information is automatically shared between all engine instances.

A. Linking Policy to Goals and Context

The core of APEX is a lightweight engine for execution of policies. It allows you to specify logic as a policy, logic that you can adapt on the fly as your system executes. The APEX policies you design can be really simple, with a single snippet of logic, or can be very complex, with many states and tasks. APEX policies can even be designed to self-adapt at execution time; each policy can adapt in real time to business/domain goals and context, as shown in Fig.1.

B. The Policy Model

The Adaptive Policy Engine in APEX runs your policies. These policies are triggered by incoming events. The logic of the policies executes and produces a response event. The Incoming Context on the incoming event and the Outgoing Context on the outgoing event are simply the fields and attributes of the event. You design the policies that APEX executes and the trigger and action events that your policies accept and produce. Events are fed in and sent out as JSON or XML events over Kafka, a Websocket, a file or named pipe, or even standard input. APEX can also be accessed programmatically, e.g. in an application, using its Java API.
You design a policy as a state chain, with each state being fed by the state before (see Fig.2). The simplest policy has one state. The four-state MEDA (Match Establish Decide Act) and the three-state ECA (Event Condition Action) policy state models are explicitly supported. APEX is fully distributed. You can decide how many APEX engine instances to run for your application and on which real or virtual hosts to run them.

In APEX, you define the context policies use and what the scope of that context is. Context is the state information and data used by policies. Policy Context is private to a particular policy and is accessible only to the APEX engines running that policy. Global Context is available to all policies. External Context is read-only context, such as weather or topology information, provided by other systems. APEX keeps context coordinated across all running instances of a policy. If a policy in an APEX engine changes a context instance value, that value is available to all other APEX engines using that context instance. APEX manages context distribution, locking, and persistence.

C. The Policy Ecosystem

The subsystems in APEX are shown in Fig.3:

- **AP-AUTH**: policy authoring (programmatic, CLI editor, REST editor, Xtext DSL editor)
- **AP-DEP**: policy deployment to one or more engines
- **AP-EN**: the actual policy engine (fully featured, multi-threaded engine with various configuration options)
- **AP-CTX**: adapters and collectors for external context information (distributed context across engines, VMs, hosts, networks)
- **AP-KB**: policy metadata knowledge base.

The APEX engine (AP-EN) can be included in an application as a Java library, as a microservice running in a Docker container, or as a stand-alone service for integration into your system. APEX also includes a web-based policy editor (AP-AUTH) and a policy management console used to deploy and keep track of the state of policies and context in policies. Context handling (AP-CTX) is integrated into the APEX deployment (AP-DEP) as a servlet running under a web framework (e.g. Apache Tomcat).

III. THE APEX DEMO

The proposed demo shows the following APEX features:

- **Policy authoring**: creating policies for different policy models using CLI, REST and XText (DSL) based editors
- **Policy deployment**: deploying a created policy to a running APEX engine (without restart)
- **Policy execution**: real and simulated policy execution for the three use cases below
- **Policy monitoring**: monitoring of policy execution KPIs using provided REST GUI applications

The demo illustrates the capabilities and power of APEX policies with the following use cases:

- VPN/Video closed loop control
- Context-aware policy
- International sales policy realization

The implementations of the features and use cases above are on Github [8].

ACKNOWLEDGMENTS

This work is partly funded by the European Commission via the ARCFIRE project (Grant 687871) under the H2020 program.

REFERENCES