For keycloak FAPI-SIG

Oct 2020

Client Policies Practical Guide

PROPOSED DRAFT

Preface

This document describes Client Policy Basics – the framework part of Client Policies

- JIRA Ticket : KEYCLOAK-14189 Client Policy : Basics
- Design document : Client Policies
- Pull Request : #7104 KEYCLOAK-14189 Client Policy : Basics

Client Policies are introduced to realize security profiles like FAPI and OAuth2 BCP in unified and extensible manner.

The intended reader is the following.

- Implementer of security profiles using Client Policies.
- Contributor to Client Policies itself.

Table of Contents

Benefits

Benefits

Client Policies can realize security profiles like FAPI and OAuth2 BCP in unified and extensible manner. Its benefits are as follow.

- Usability

  It can improve messy client settings.

- Code Maintainability/Extensibility/Availability

  It can improve the readability of endpoint classes.

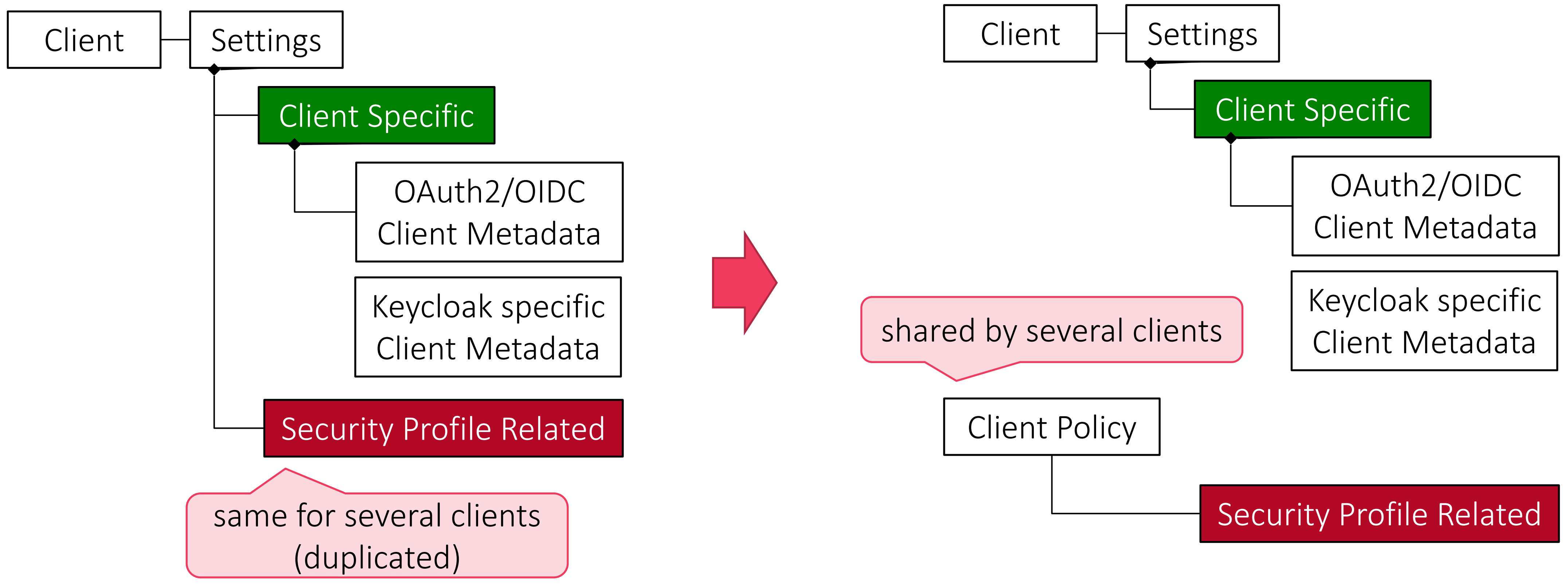
- Backward Compatibility

  It can realize what the current Client Registration Policies do

Benefits - Usability

- Usability

Security profile related client settings can be moved from Admin Console's client settings UI.

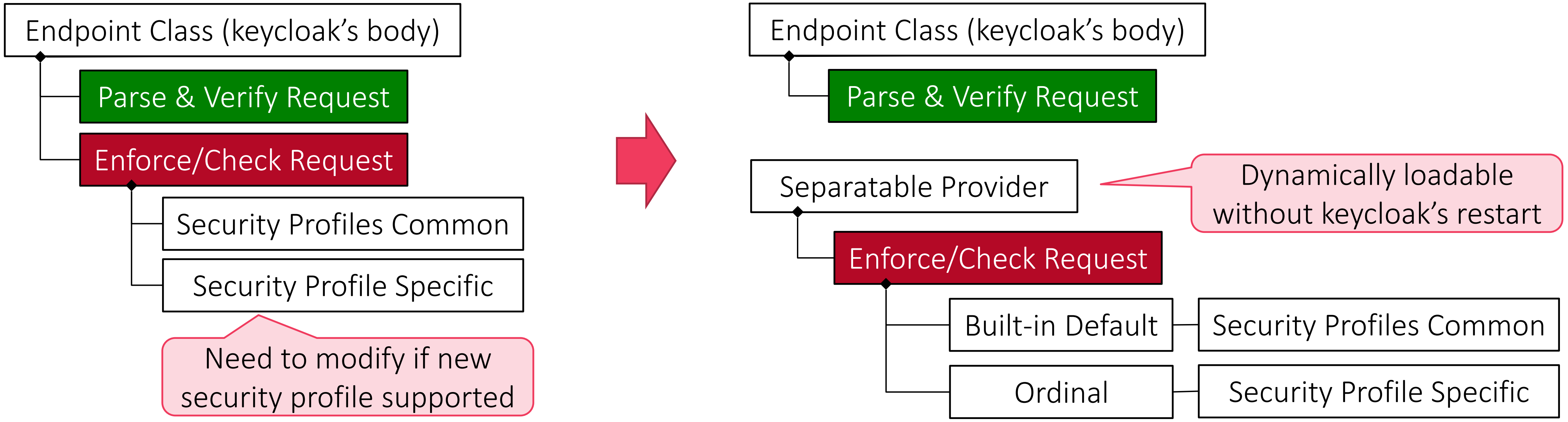

Client
Settings
Client Specific
OAuth2/OIDC Client Metadata
Keycloak specific Client Metadata
Security Profile Related
same for several clients (duplicated)
Client
Settings
Client Specific
OAuth2/OIDC Client Metadata
Keycloak specific Client Metadata
shared by several clients
Client Policy
Security Profile Related

PROPOSED DRAFT

Benefits - Code Maintainability/Extensibility/Availability

- Code Maintainability/Extensibility/Availability

Security profile related hardcoded codes can be moved from endpoint classes to separatable providers. Keycloak need not restart when introducing new policies.


Endpoint Class (keycloak's body)
Parse & Verify Request
Enforce/Check Request
Security Profiles Common
Security Profile Specific
Need to modify if new security profile supported
Endpoint Class (keycloak's body)
Parse & Verify Request
Separatable Provider
Dynamically loadable without keycloak's restart
Enforce/Check Request
Built-in Default
Security Profiles Common
Ordinal
Security Profile Specific

PROPOSED DRAFT

Benefits - Backward Compatibility

- Backward Compatibility

  It can realize what the current Client Registration Policies do

- Provider :

  Re-implemented as Executor of Client Policies

- UI :

  Re-realized by Client Policies' Admin Console UI

- Persistent Entity :

  Migrated to Client Policies' ComponentModel by liquibase

Scope

Scope

This document covers the followings :

■ JIRA Ticket : KEYCLOAK-14189 Client Policy : Basics

■ Design document : Client Policies

■ Pull Request : #7104 KEYCLOAK-14189 Client Policy : Basics

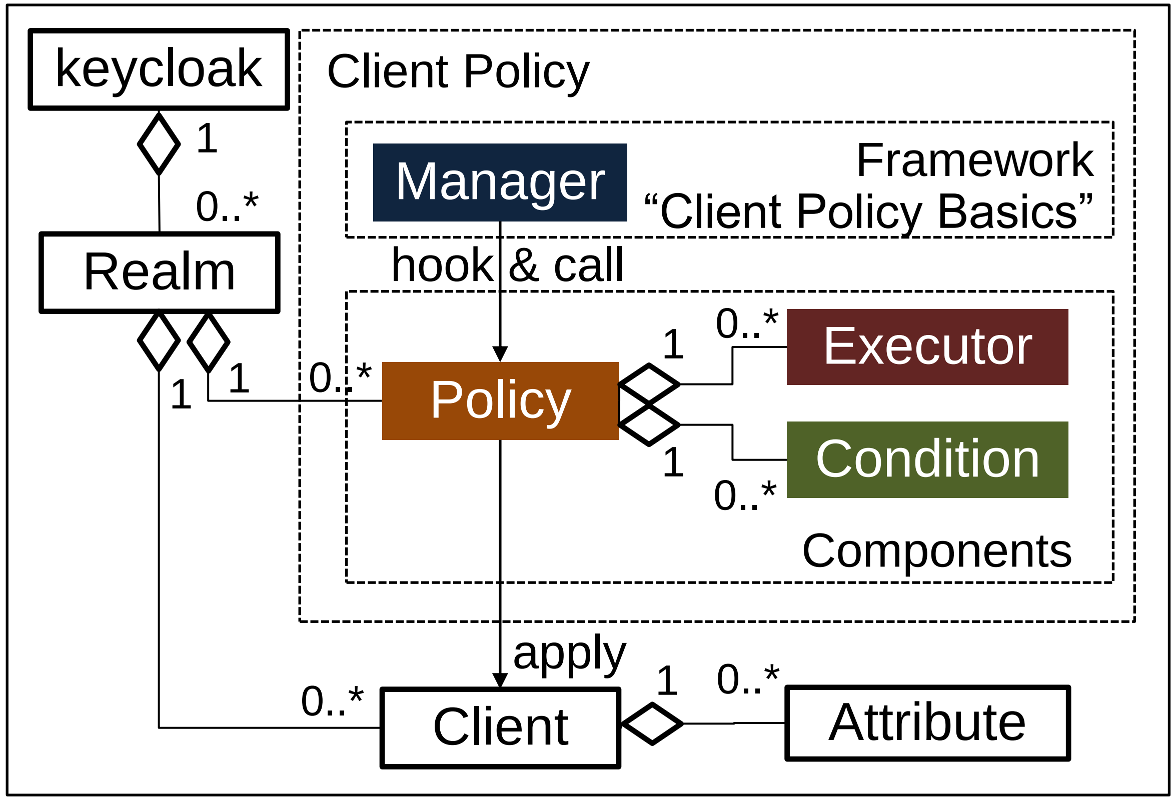
Especially, this document does not cover the followings :

□ JIRA sub tickets of KEYCLOAK-13933 Client Policies other than Client Policy : Basics.

- Conditions/Executors Implementation for FAPI-RW security profile support
- Admin Console UI
- Pre-set Policies
- Client Registration Policies Migration

Mechanism

Layout



Basically, Client Policies consists of two parts.

- Framework

  Load & run components.

  Implemented on keycloak's body.

- Components

  Security Profile's process.

  Determine which security profile applied to which client.

PROPOSED DRAFT

Elements

[Components]

- Executor
- Condition
- Policy

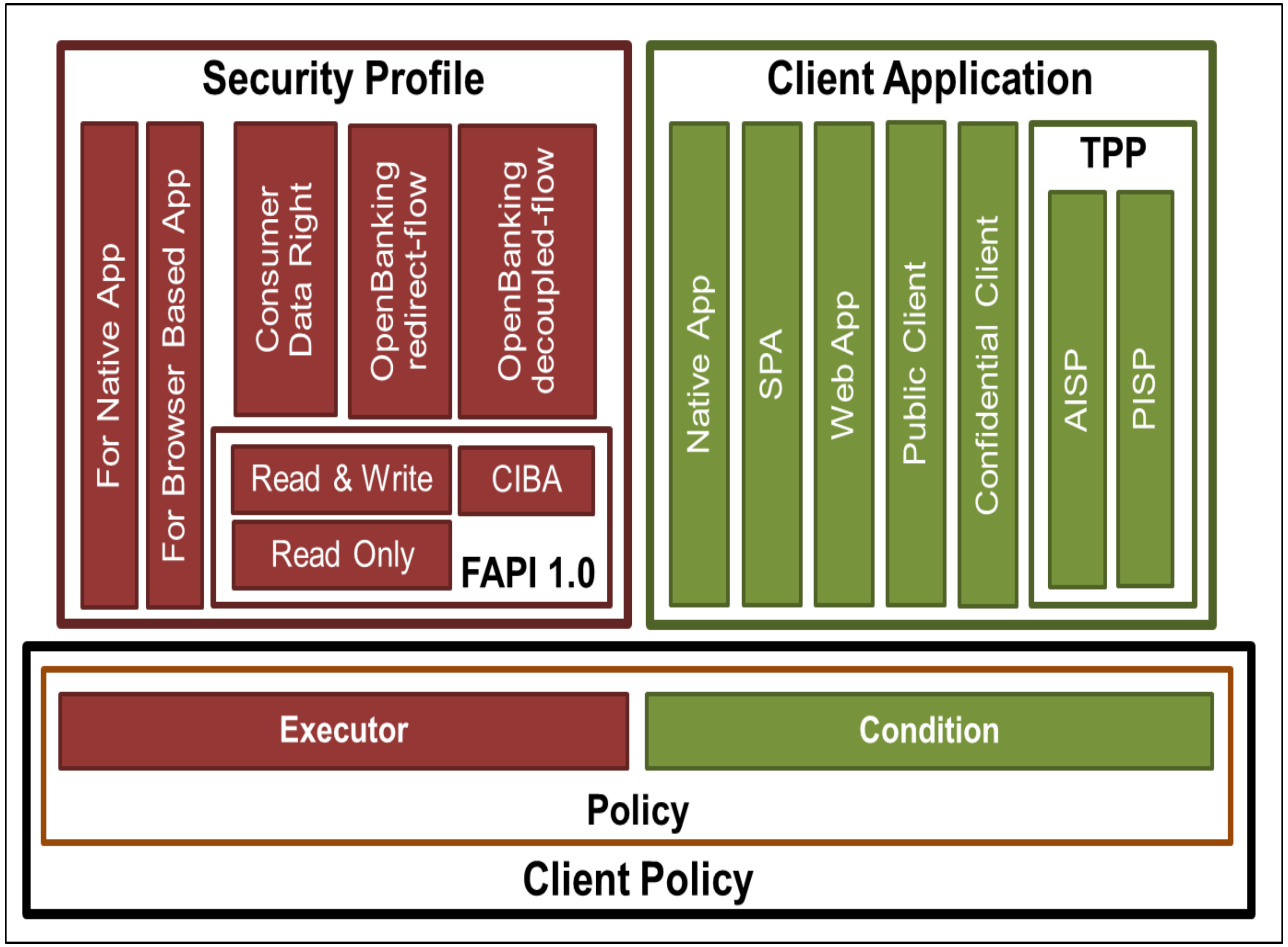
[Framework]

- Event
- Context
- Manager

[Misc]

- Vote
- Exception
- Logger
- Test : ClientPolicyBasicsTest

Components



- Executor : what to do

  Implement security profile's process

- Condition : to which client

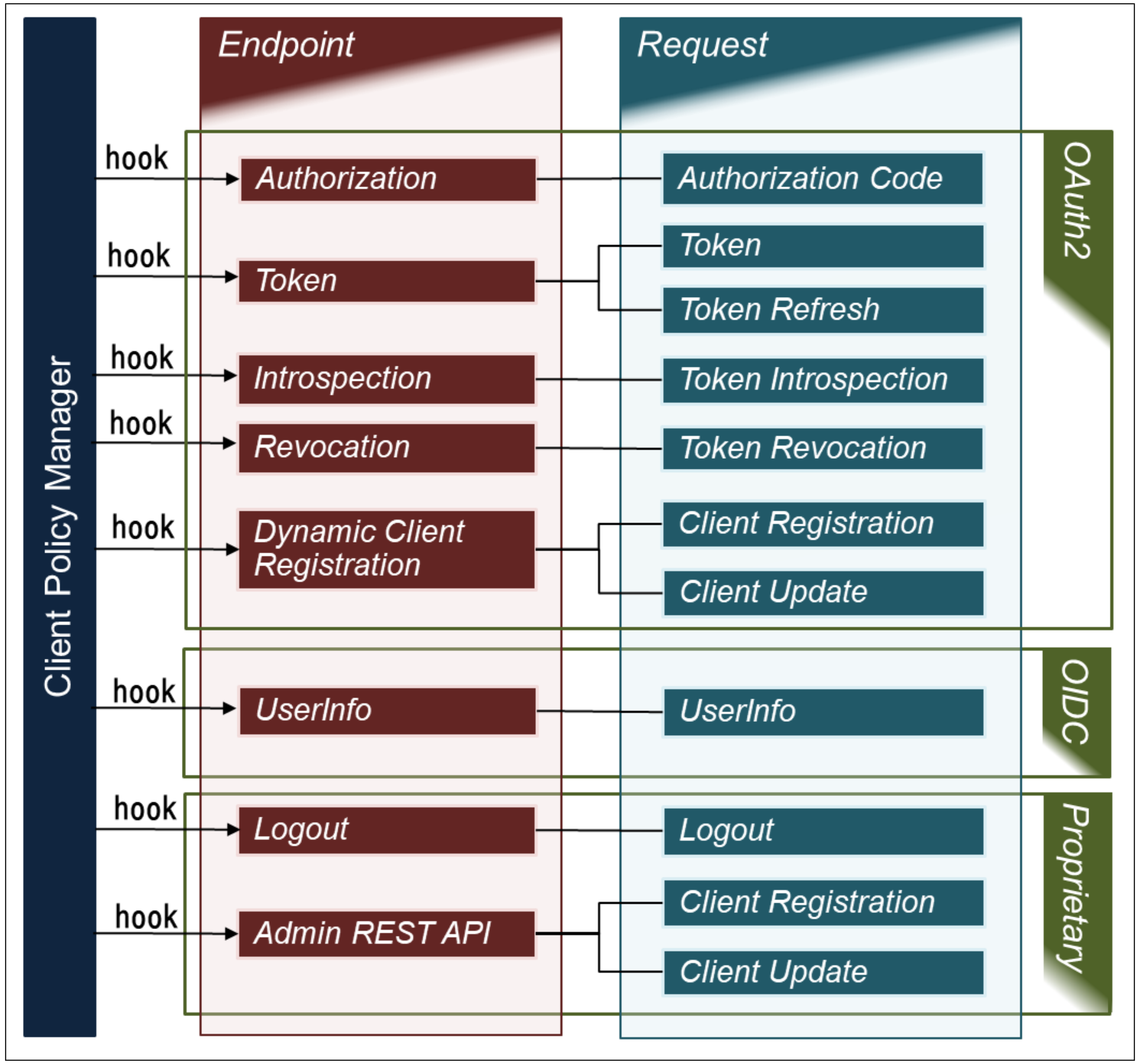
  Determine which security profile applied to which client.

- Policy : what & which client

  Manage a pair of Executor and Condition.

PROPOSED DRAFT

Framework



- Event : what kind of request arrives

  Indicate which type of request arrives on which endpoint.

- Context : what request in detail

  Contain the context data of the received request.

- Manager : monitor everything

  Dependent on Event and Context, evaluate Policy's Condition to determine whether Policy's Executor is applied to the client sending the request. If so, execute Policy's Executor.

PROPOSED DRAFT

Misc

- Vote

  The return value of evaluating Condition

  “YES” : the client satisfies this Condition

  “NO” : the client does not satisfy this Condition

  “ABSTAIN” : skipping the evaluation of this Condition
- Exception

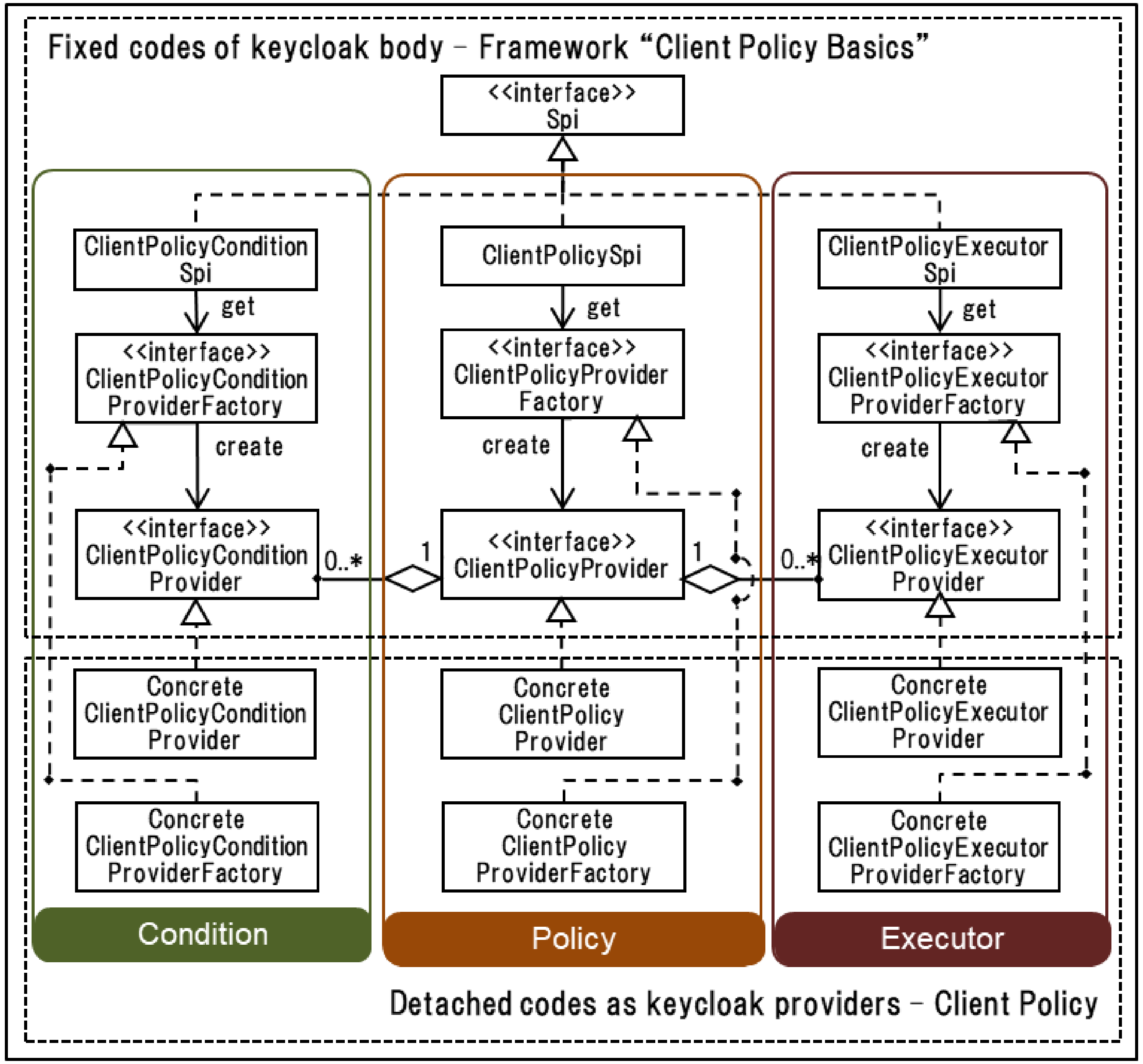
  Showing errors occurred when evaluating Condition and executing Executor.
- Logger

  Logging events occurred when evaluating Condition and executing Executor.
- Test : ClientPolicyBasicsTest

  Testing Client Policies feature by Arquillian Integration Test framework.

Internals

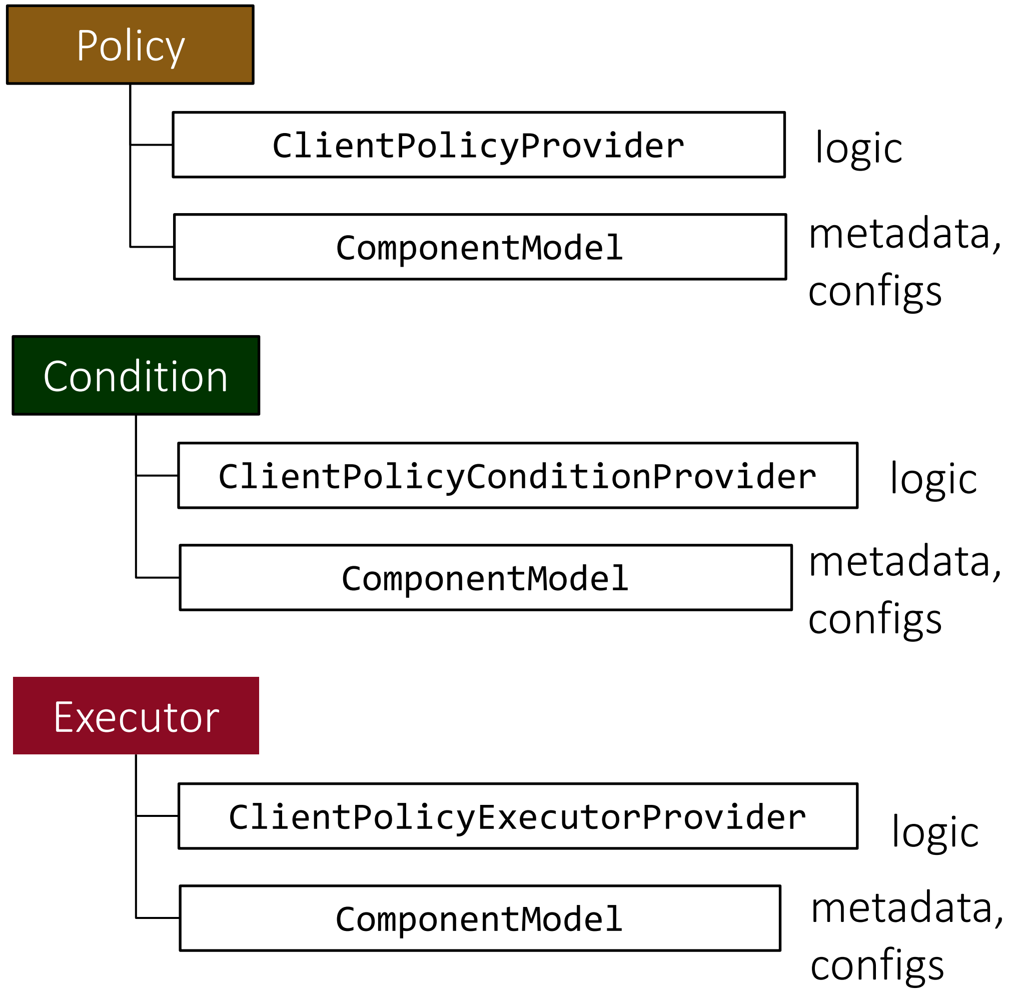
Implementation



Fixed codes of keycloak body – Framework "Client Policy Basics"
<<interface>> Spi
ClientPolicyCondition Spi
get
<<interface>> ClientPolicyCondition ProviderFactory
create
<<interface>> ClientPolicyCondition Provider
0..*
1
ClientPolicySpi
get
<<interface>> ClientPolicyProvider Factory
create
<<interface>> ClientPolicyProvider
1
0..*
ClientPolicyExecutor Spi
get
<<interface>> ClientPolicyExecutor ProviderFactory
create
<<interface>> ClientPolicyExecutor Provider
Concrete ClientPolicyCondition Provider
Concrete ClientPolicyCondition ProviderFactory
Concrete ClientPolicy Provider
Concrete ClientPolicy ProviderFactory
Concrete ClientPolicyExecutor Provider
Concrete ClientPolicyExecutor ProviderFactory
Condition
Policy
Executor
Detached codes as keycloak providers – Client Policy

- Fixed codes of keycloak body
  - Client Policy Basics (Framework)
  - Interface of Policy/Condition/Executor
- Detached codes as keycloak providers
  - Concrete Policy/Condition/Executor

Policy/Condition/Executor : Persisted Metadata & Configs



- Each of Policy, Condition and Executor persist their metadata and configs on `ComponentModel` class that is generally used to accommodate any kind of provider's metadata and configs.

[Metadata]

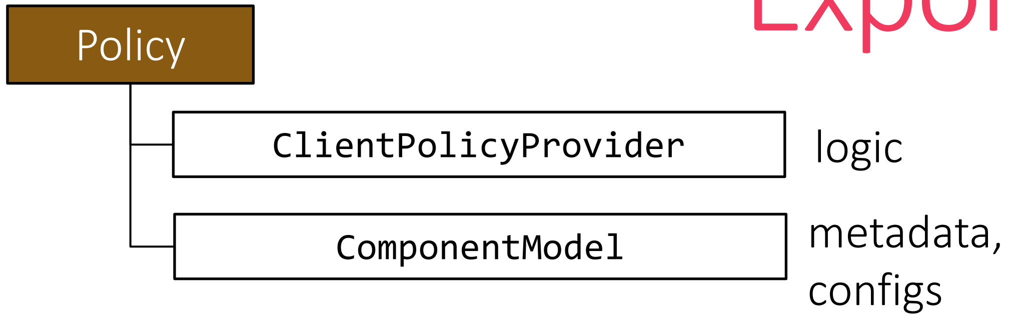
- ID : Component's instance ID
- Name : Component's instance name
- Provider ID : Component's provider ID defined by concrete provider factory class.

[Configs]

Depending on each components.

PROPOSED DRAFT

Policy: Persisted Metadata & Configs Exported JSON Representation

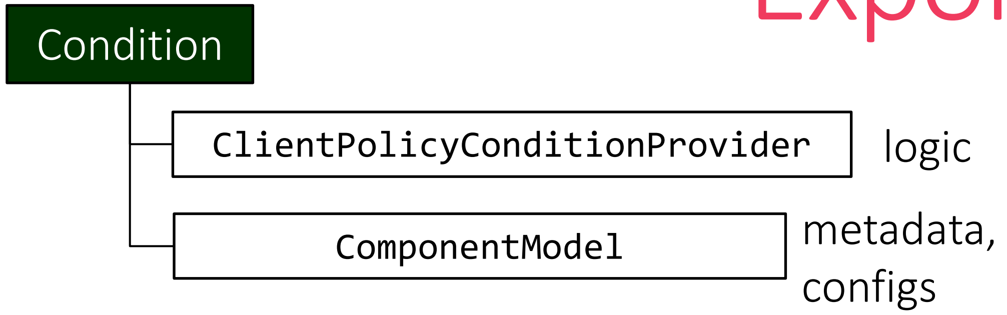


```
 "components": {
...
    "org.keycloak.services.clientpolicy.ClientPolicyProvider" : [ {
          "name" : "MyPolicy",
          "providerId" : "client-policy-provider",
          "subComponents" : { },
          "config" : {
            "client-policy-executor-ids" : [ "abad6ee9-6f4b-4e12-bc23-a5da2fdb6abf",
                                             "47e818e3-f0d8-4a90-ace7-1a8b2362cec7" ],
            "client-policy-condition-ids" : [ "047eb7c3-ba5f-45bc-9e27-9ca0d3ae5146" ]
          }
        } ],
...
```

Accommodating executors' ids

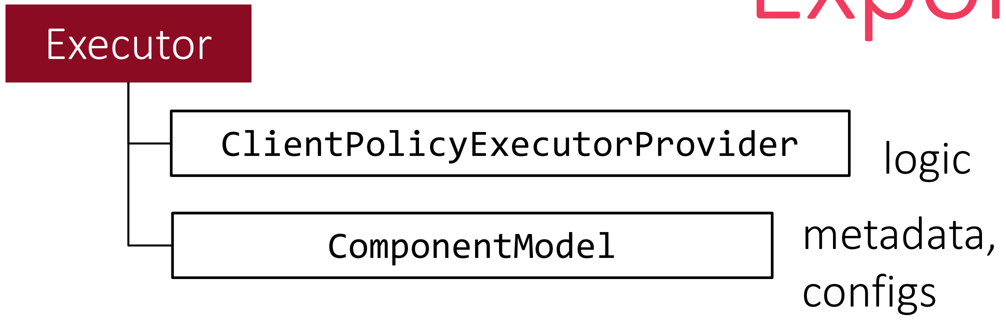
Accommodating conditions' ids

Condition : Persisted Metadata & Configs Exported JSON Representation



```
 "components": {
…
    "org.keycloak.services.clientpolicy.condition.ClientPolicyConditionProvider" : [ {
            "id" : "047eb7c3-ba5f-45bc-9e27-9ca0d3ae5146",
            "name" : "ClientRolesCondition",
            "providerId" : "clientroles-condition",
            "subComponents" : { },
            "config" : {
              "roles" : [ "sample-client-role" ]
            }
          } ]
…
```

Executor : Persisted Metadata & Configs Exported JSON Representation



```
 "components": {
…
    "org.keycloak.services.clientpolicy.executor.ClientPolicyExecutorProvider" : [ {
            "id" : "47e818e3-f0d8-4a90-ace7-1a8b2362cec7",
            "name" : "SecureRequestObjectExecutor",
            "providerId" : "secure-reqobj-executor",
            "subComponents" : { },
            "config" : { }
          },
,…
```

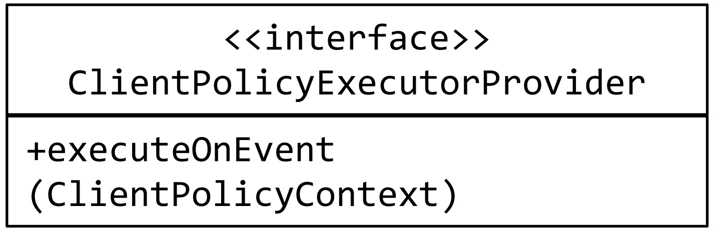
Condition : How to Implement

```
<<interface>>
ClientPolicyConditionProvider
-----------------------------
+applyPolicy(ClientPolicyContext)
```

- Implement **`applyPolicy`** method.
- Implement codes on each Event.
- Dependent on Event, **`ClientPolicyContext`** argument is casted into the corresponding Context class (p.30,31).
- Return `ABSTAIN` if the Event is not evaluated.

```java
@Override
    public ClientPolicyVote applyPolicy(ClientPolicyContext context) throws ClientPolicyException {
        switch (context.getEvent()) {
        case REGISTER:
        case UPDATE:
            if (isAuthMethodMatched((ClientUpdateContext)context)) return ClientPolicyVote.YES;
            return ClientPolicyVote.NO;
        default:
            return ClientPolicyVote.ABSTAIN;
        }
    }
```

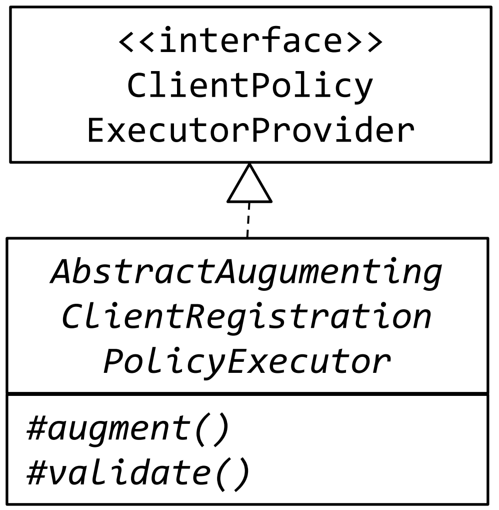
Executor : How to Implement

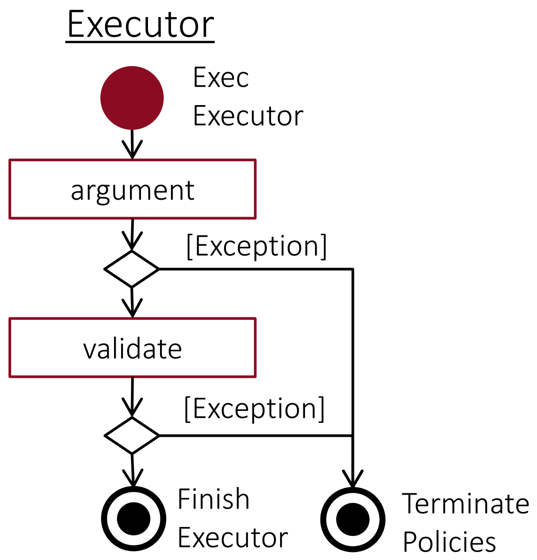


- Implement **`executeOnEvent`** method.
- Implement codes on each Event.
- Dependent on Event, **`ClientPolicyContext`** argument is casted into the corresponding Context class (p.30,31).
- Throw **`ClientPolicyException`** the client's request is prohibited due to not complying with the security profile implemented by the executor.

```java
@Override
public void executeOnEvent(ClientPolicyContext context) throws ClientPolicyException {
    switch (context.getEvent()) {
        case AUTHORIZATION_REQUEST:
            AuthorizationRequestContext authorizationRequestContext = (AuthorizationRequestContext)context;
            executeOnAuthorizationRequest(authorizationRequestContext.getparsedResponseType(),
                authorizationRequestContext.getAuthorizationEndpointRequest(),
                authorizationRequestContext.getRedirectUri(),
                authorizationRequestContext.getRequestParameters());
            break;
        default:
            return;
    }
 }
```

Augmenting Executor for Client Registration/Update

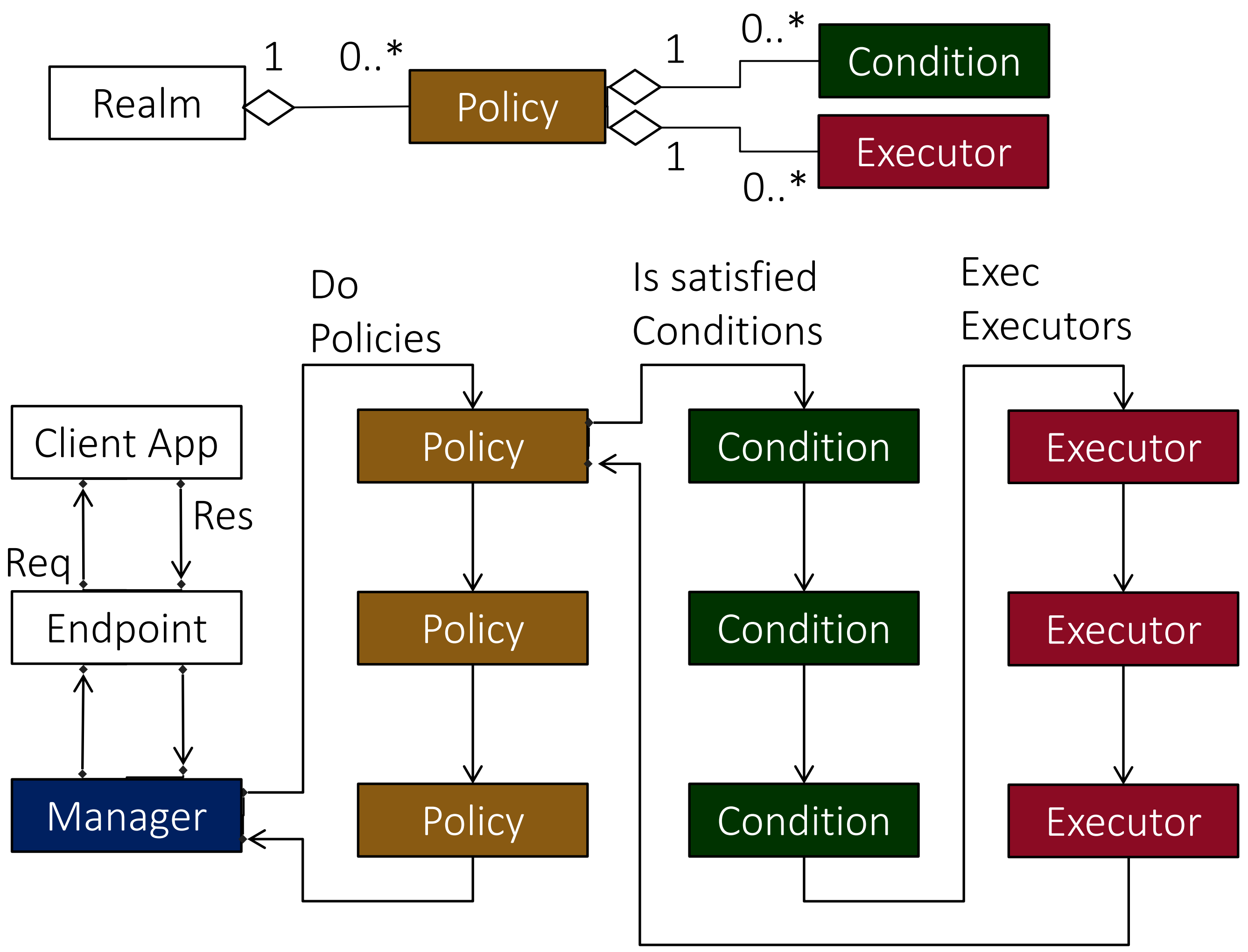




- At argument phase, some security profile related settings are enforced.
- At validate phase, some security profile related settings are checked to confirm whether there are appropriate or not.
- Subclasses overrides `argument` and `validate` method.

```
@Override
   public void executeOnEvent(ClientPolicyContext context) throws ClientPolicyException {
       switch (context.getEvent()) {
       case REGISTER:
       case UPDATE:
           ClientUpdateContext clientUpdateContext = (ClientUpdateContext)context;
           augment(clientUpdateContext.getProposedClientRepresentation());
           validate(clientUpdateContext.getProposedClientRepresentation());
           break;
       default:
           return;
       }
   }
```

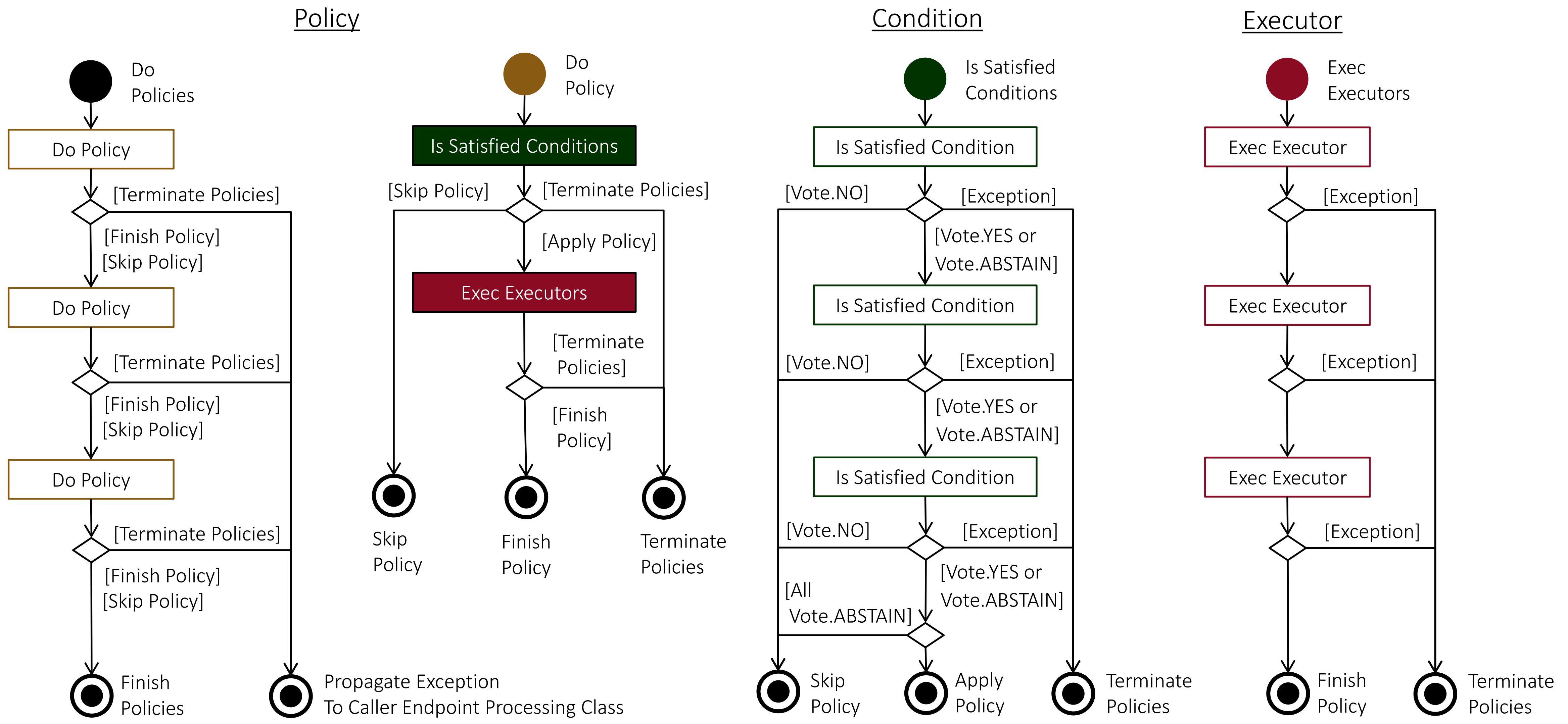
Policy/Condition/Executor : Cardinality, Duplication, Order of Execution



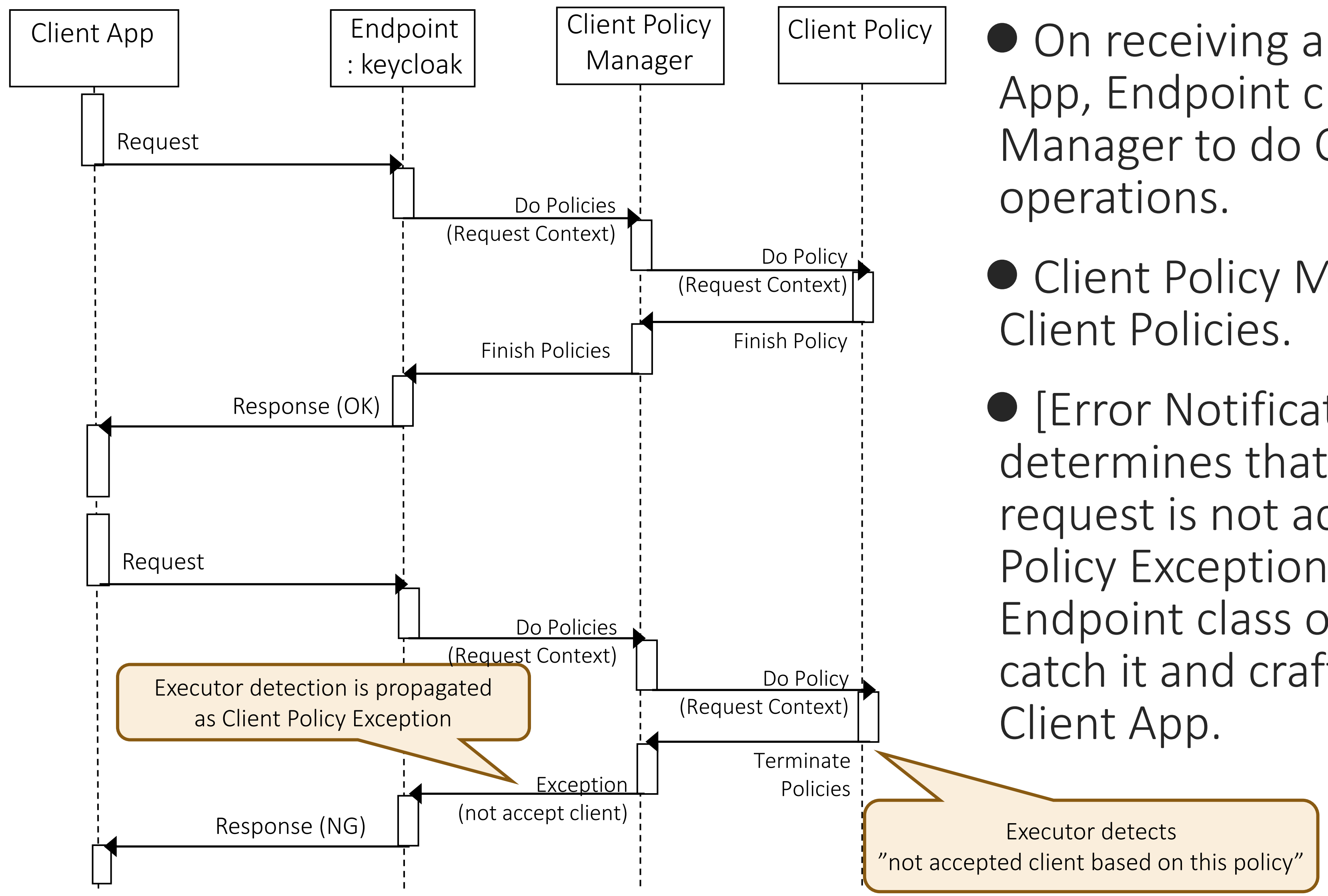
- [Cardinality] Realm can accommodate multiple Policies. The number of Policies is unlimited.
- [Cardinality] Policy can accommodate multiple Conditions and Executors. The number of Conditions/Executors are unlimited.
- [Duplication] It is allowed to duplicate Policy/Condition/Executor.
- [Order of Execution] We can not specify the order of execution of Policies/Conditions/Executors.

PROPOSED DRAFT

Client Policies Evaluation Activity Diagram



Client Policies Evaluation Flow Diagram

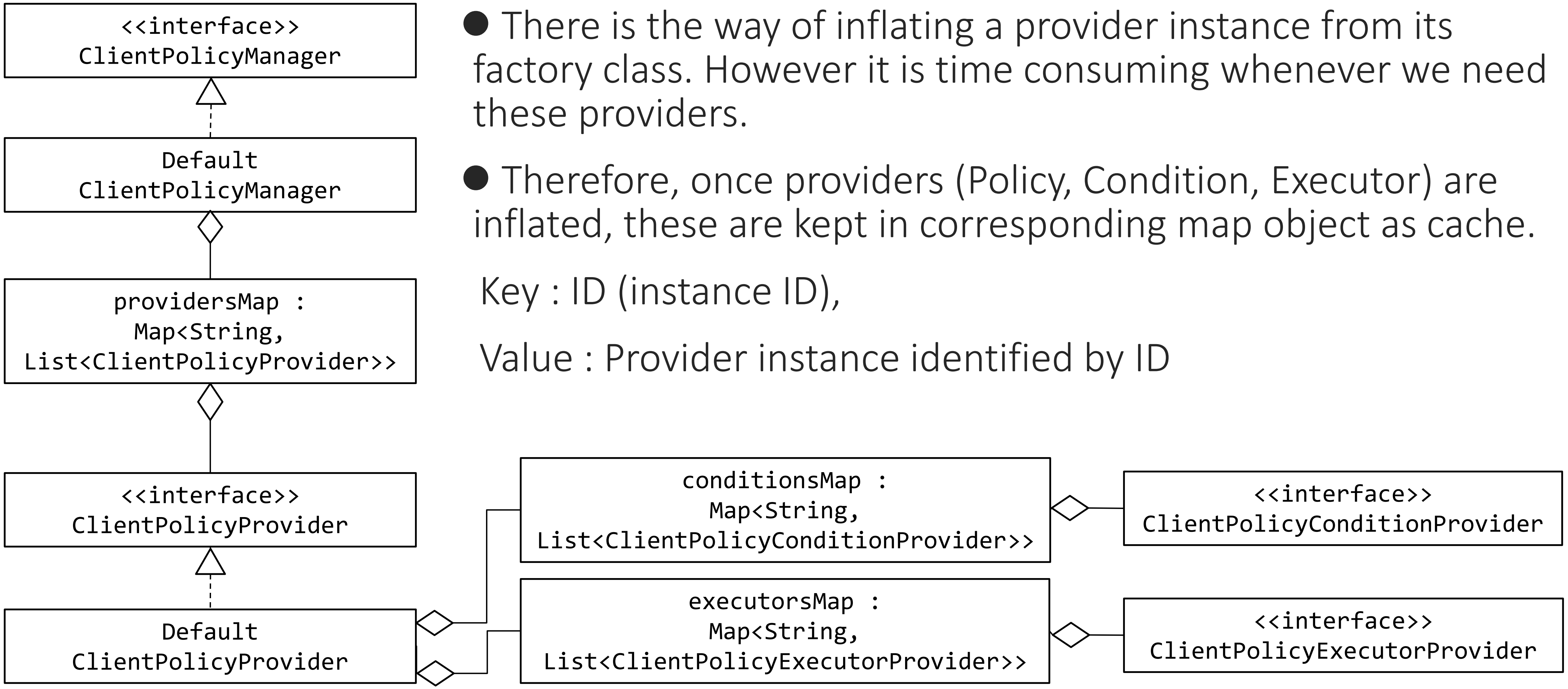


Client App
Endpoint : keycloak
Client Policy Manager
Client Policy
Request
Do Policies (Request Context)
Do Policy (Request Context)
Finish Policy
Finish Policies
Response (OK)
Request
Do Policies (Request Context)
Do Policy (Request Context)
Terminate Policies
Exception (not accept client)
Response (NG)
Executor detection is propagated as Client Policy Exception
Executor detects ”not accepted client based on this policy”

- On receiving a request from Client App, Endpoint class asks Client Policy Manager to do Client Policies operations.
- Client Policy Manager evaluate all Client Policies.
- [Error Notification] If some Executor determines that this Client App’s request is not acceptable, throw Client Policy Exception to notify caller Endpoint class of it. Endpoint class catch it and crafts am error response to Client App.

PROPOSED DRAFT

Performance - Provider Cache



<<interface>> ClientPolicyManager
Default ClientPolicyManager
providersMap : Map<String, List<ClientPolicyProvider>>
<<interface>> ClientPolicyProvider
Default ClientPolicyProvider
conditionsMap : Map<String, List<ClientPolicyConditionProvider>>
executorsMap : Map<String, List<ClientPolicyExecutorProvider>>
<<interface>> ClientPolicyConditionProvider
<<interface>> ClientPolicyExecutorProvider

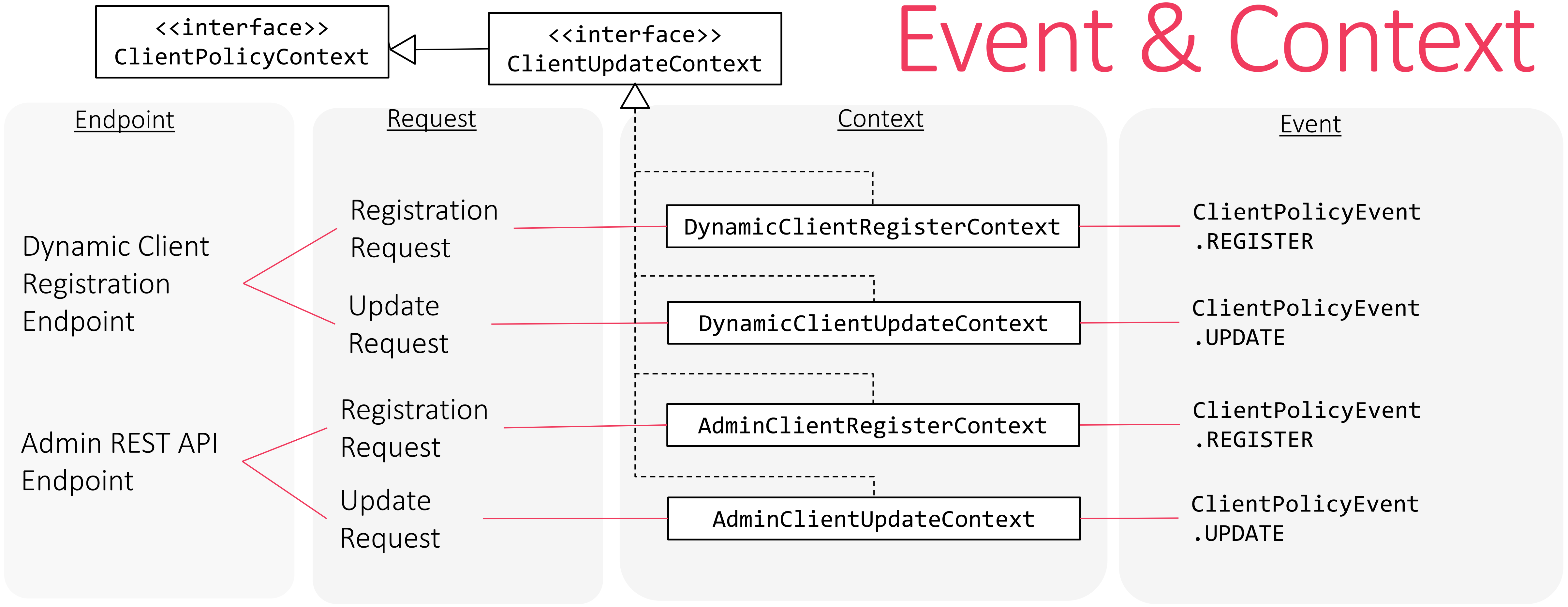
- There is the way of inflating a provider instance from its factory class. However it is time consuming whenever we need these providers.
- Therefore, once providers (Policy, Condition, Executor) are inflated, these are kept in corresponding map object as cache.

Key : ID (instance ID),

Value : Provider instance identified by ID

PROPOSED DRAFT

Event & Context

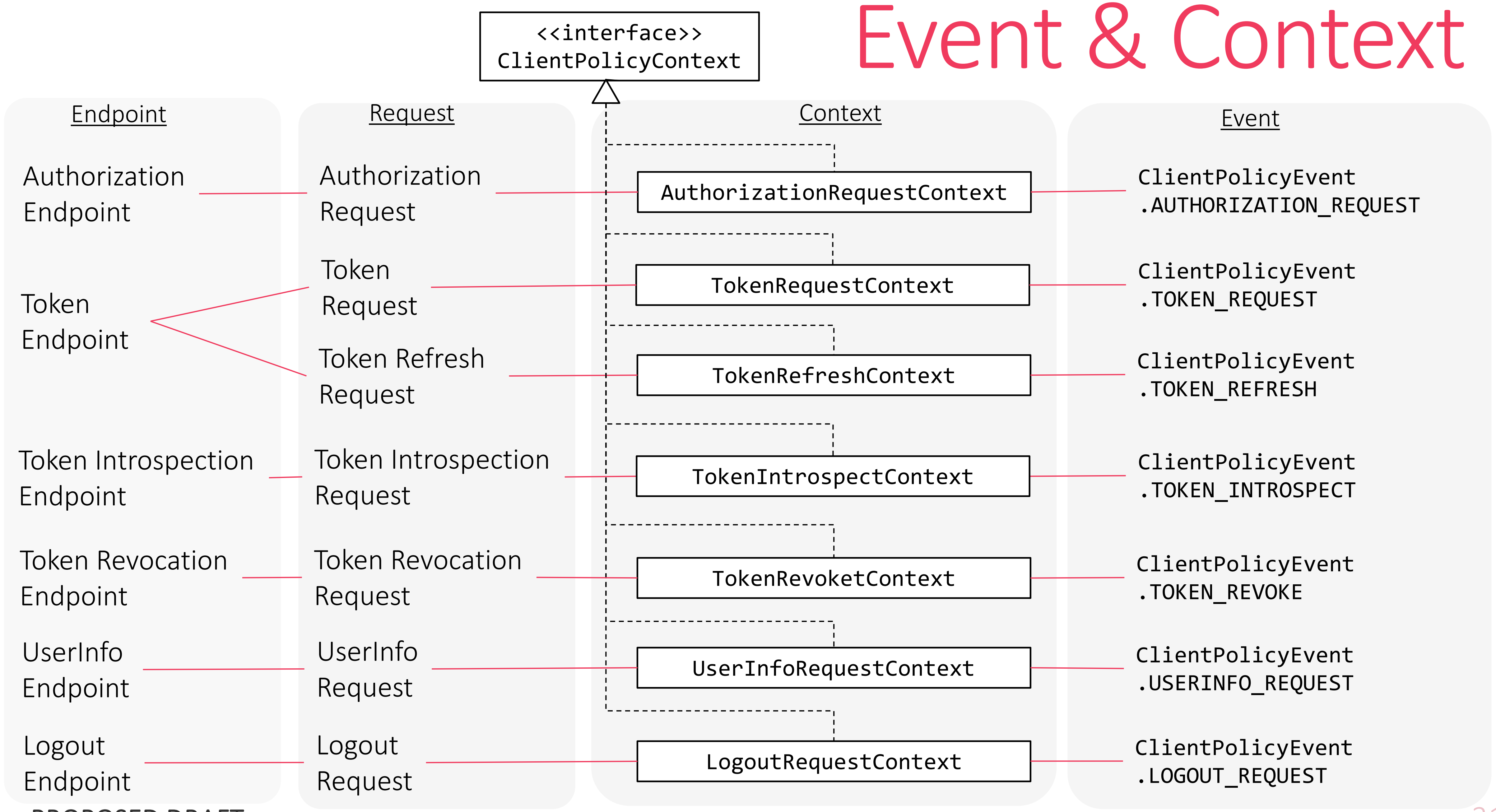


<<interface>> ClientPolicyContext
<<interface>> ClientUpdateContext
Endpoint
Request
Context
Event
Dynamic Client Registration Endpoint
Admin REST API Endpoint
Registration Request
Update Request
Registration Request
Update Request
DynamicClientRegisterContext
DynamicClientUpdateContext
AdminClientRegisterContext
AdminClientUpdateContext
ClientPolicyEvent .REGISTER
ClientPolicyEvent .UPDATE
ClientPolicyEvent .REGISTER
ClientPolicyEvent .UPDATE

{ Endpoint, Request } → Context → Event

- A pair of Endpoint and the type of Request that the Client sends to the Endpoint corresponds to the dedicated Context.
- Context corresponds to the dedicated Event.

Event & Context



PROPOSED DRAFT

Exception

- [Exception] Following Exception class is used for Client Policies related codes :

server-spi/src/main/java/org/keycloak/services/clientpolicy/ClientPolicyException.java

- [Usage of this exception in Executor] This exception is used to inform the caller Endpoint class of the client's request is prohibited due to not complying with the security profile implemented by the executor.

```
...
    // check whether "aud" claim exists
    List<String> aud = new ArrayList<String>();
    JsonNode audience = requestObject.get("aud");
    if (audience == null) {
        ClientPolicyLogger.log(logger, "aud claim not incuded.");
        throw new ClientPolicyException(INVALID_REQUEST_OBJECT,
                                        "Missing parameter : aud");
    }
...
```

Logging

- [Logger] Following Logger class is used for Client Policies related codes :

services/src/main/java/org/keycloak/services/clientpolicy/ClientPolicyLogger.java

- [Log Level] This logger’s log level is “trace”.

- [LoggingControl on Arquillian Integration Test] When running Arquillian Integration Test (ClientPolicyBasicsTest), you can control this logging by modifying the following properties file.

testsuite/integration-arquillian/tests/base/src/test/resources/log4j.properties

```
…
# log4j.logger.org.keycloak.services.clientpolicy=trace
# log4j.logger.org.keycloak.testsuite.clientpolicy=trace
…
```

Tests

- [Arquillian Integration Test Class] Following test class is used :

testsuite/integration-arquillian/tests/base/src/test/java/org/keycloak/testsuite/client/ClientPolicyBasicsTest.java

- [Test Run Command] The command for running this test is as follows :

```
# Build
mvn clean install -DskipTests –Pdistribution

# Build testsuite
mvn clean install -Pauth-server-wildfly -DskipTests -f testsuite/pom.xml

# Run base tests
mvn -f testsuite/integration-arquillian/tests/base/pom.xml -Pauth-server-wildfly test
-Dtest=org.keycloak.testsuite.client.ClientPolicyBasicsTest –Dkeycloak.logging.level=info
```

PROPOSED DRAFT

Client Policy Basics Classes

[1] Tech Preview Setting

common/src/main/java/org/keycloak/common/Profile.java

[2] Unit Test : Tech Preview Setting

common/src/test/java/org/keycloak/common/ProfileTest.java

[3] Policy : Provider - Interface

server-spi-private/src/main/java/org/keycloak/services/clientpolicy/ClientPolicyProvider.java

[4] Policy : Provider Factory - Interface

server-spi-private/src/main/java/org/keycloak/services/clientpolicy/ClientPolicyProviderFactory.java

[5] Policy : SPI

server-spi-private/src/main/java/org/keycloak/services/clientpolicy/ClientPolicySpi.java

[6] Vote

server-spi-private/src/main/java/org/keycloak/services/clientpolicy/ClientPolicyVote.java

Client Policy Basics Classes

[7] Condition : Provider - Interface

server-spi-private/src/main/java/org/keycloak/services/clientpolicy/condition/ClientPolicyConditionProvider.java

[8] Condition : Provider Factory - Interface

server-spi-private/src/main/java/org/keycloak/services/clientpolicy/condition/ClientPolicyConditionProviderFactory.java

[9] Condition : SPI

server-spi-private/src/main/java/org/keycloak/services/clientpolicy/condition/ClientPolicyConditionSpi.java

[10] Executor : Provider - Interface

server-spi-private/src/main/java/org/keycloak/services/clientpolicy/executor/ClientPolicyExecutorProvider.java

[11] Executor : Provider Factory - Interface

server-spi-private/src/main/java/org/keycloak/services/clientpolicy/executor/ClientPolicyExecutorProviderFactory.java

[12] Executor : SPI

server-spi-private/src/main/java/org/keycloak/services/clientpolicy/executor/ClientPolicyExecutorSpi.java

Client Policy Basics Classes

[13] Policy/Condition/Executor : SPI Registration

server-spi-private/src/main/resources/META-INF/services/org.keycloak.provider.Spi

[14] Keycloak Session - Interface

server-spi/src/main/java/org/keycloak/models/KeycloakSession.java

Provides Client Policy Manager.

[15] Context - Interface

server-spi/src/main/java/org/keycloak/services/clientpolicy/ClientPolicyContext.java

[16] Event

server-spi/src/main/java/org/keycloak/services/clientpolicy/ClientPolicyEvent.java

[17] Exception

server-spi/src/main/java/org/keycloak/services/clientpolicy/ClientPolicyException.java

[18] Manager - Interface

server-spi/src/main/java/org/keycloak/services/clientpolicy/ClientPolicyManager.java

Client Policy Basics Classes

[19] Endpoint – Authorization Endpoint

services/src/main/java/org/keycloak/protocol/oidc/endpoints/AuthorizationEndpoint.java

[20] Endpoint – Logout Endpoint

services/src/main/java/org/keycloak/protocol/oidc/endpoints/LogoutEndpoint.java

[21] Endpoint – Token Endpoint

services/src/main/java/org/keycloak/protocol/oidc/endpoints/TokenEndpoint.java

[22] Endpoint – Token Introspection Endpoint

services/src/main/java/org/keycloak/protocol/oidc/endpoints/TokenIntrospectionEndpoint.java

[23] Endpoint – Token Revocation Endpoint

services/src/main/java/org/keycloak/protocol/oidc/endpoints/TokenRevocationEndpoint.java

[24] Endpoint – UserInfo Endpoint

services/src/main/java/org/keycloak/protocol/oidc/endpoints/UserInfoEndpoint.java

Client Policy Basics Classes

[25] Keycloak Session – Default Implementation

services/src/main/java/org/keycloak/services/DefaultKeycloakSession.java

Provides Client Policy Manager.

[26] Context – Client Registration Request by Admin REST API

services/src/main/java/org/keycloak/services/clientpolicy/AdminClientRegisterContext.java

[27] Context – Client Update Request by Admin REST API

services/src/main/java/org/keycloak/services/clientpolicy/AdminClientUpdateContext.java

[28] Context – Authorization Request

services/src/main/java/org/keycloak/services/clientpolicy/AuthorizationRequestContext.java

[29] Logger

services/src/main/java/org/keycloak/services/clientpolicy/ClientPolicyLogger.java

[30] Context – General Client Registration/Update Request Interface

services/src/main/java/org/keycloak/services/clientpolicy/ClientUpdateContext.java

Client Policy Basics Classes

[31] Manager – Default Implementation

services/src/main/java/org/keycloak/services/clientpolicy/DefaultClientPolicyManager.java

[32] Policy – Default Implementation

services/src/main/java/org/keycloak/services/clientpolicy/DefaultClientPolicyProvider.java

[33] Policy Factory – Default Implementation

services/src/main/java/org/keycloak/services/clientpolicy/DefaultClientPolicyProviderFactory.java

[34] Context – Dynamic Client Registration Request

services/src/main/java/org/keycloak/services/clientpolicy/DynamicClientRegisterContext.java

[35] Context – Dynamic Client Update Request

services/src/main/java/org/keycloak/services/clientpolicy/DynamicClientUpdateContext.java

[36] Context – Logout Request

services/src/main/java/org/keycloak/services/clientpolicy/LogoutRequestContext.java

Client Policy Basics Classes

[37] Context – Token Introspection Request

services/src/main/java/org/keycloak/services/clientpolicy/TokenIntrospectContext.java

[38] Context – Token Refresh Request

services/src/main/java/org/keycloak/services/clientpolicy/TokenRefreshContext.java

[39] Context – Token Request

services/src/main/java/org/keycloak/services/clientpolicy/TokenRequestContext.java

[40] Context – Token Revocation Request

services/src/main/java/org/keycloak/services/clientpolicy/TokenRevokeContext.java

[41] Context – UserInfo Request

services/src/main/java/org/keycloak/services/clientpolicy/UserInfoRequestContext.java

[42] Executor – Abstract Augmenting Executor for Client Registration/Update

services/src/main/java/org/keycloak/services/clientpolicy/executor/AbstractAugumentingClientRegistration
PolicyExecutor.java

Client Policy Basics Classes

[43] Executor Factory – Abstract Augmenting Executor Factory for Client Registration/Update

services/src/main/java/org/keycloak/services/clientpolicy/executor/AbstractAugumentingClientRegistrationPolicyExecutorFactory.java

[44] Endpoint – Dynamic Client Registration/Update

services/src/main/java/org/keycloak/services/clientregistration/ClientRegistrationAuth.java

[45] Endpoint – Admin REST API for Client Update

services/src/main/java/org/keycloak/services/resources/admin/ClientResource.java

[46] Endpoint – Admin REST API for Client Registration

services/src/main/java/org/keycloak/services/resources/admin/ClientsResource.java

[47] Policy Implementation Registration

services/src/main/resources/META-INF/services/org.keycloak.services.clientpolicy.ClientPolicyProviderFactory

Client Policy Basics Classes

[48] Condition – Test Provider Implementation : the way of creating/updating a client

testsuite/integration-arquillian/servers/auth-server/services/testsuite-providers/src/main/java/org/keycloak/testsuite/services/clientpolicy/condition/TestAuthnMethodsCondition.java

[49] Condition Factory – Test Provider Factory Implementation : the way of creating/updating a client

testsuite/integration-arquillian/servers/auth-server/services/testsuite-providers/src/main/java/org/keycloak/testsuite/services/clientpolicy/condition/TestAuthnMethodsConditionFactory.java

[50] Condition – Test Provider Implementation : Client Role

testsuite/integration-arquillian/servers/auth-server/services/testsuite-providers/src/main/java/org/keycloak/testsuite/services/clientpolicy/condition/TestClientRolesCondition.java

[51] Condition Factory – Test Provider Factory Implementation : Client Role

testsuite/integration-arquillian/servers/auth-server/services/testsuite-providers/src/main/java/org/keycloak/testsuite/services/clientpolicy/condition/TestClientRolesConditionFactory.java

Client Policy Basics Classes

[52] Condition – Test Provider Implementation : Raise Client Policy Exception Intentionally

testsuite/integration-arquillian/servers/auth-server/services/testsuite-providers/src/main/java/org/keycloak/testsuite/services/clientpolicy/condition/TestRaiseExeptionCondition.java

[53] Condition Factory – Test Provider Factory Implementation : Raise Client Policy Exception Intentionally

testsuite/integration-arquillian/servers/auth-server/services/testsuite-providers/src/main/java/org/keycloak/testsuite/services/clientpolicy/condition/TestRaiseExeptionConditionFactory.java

[54] Executor – Test Provider Implementation : Enforce more secure client authentication method when client registration

testsuite/integration-arquillian/servers/auth-server/services/testsuite-providers/src/main/java/org/keycloak/testsuite/services/clientpolicy/executor/TestClientAuthenticationExecutor.java

[55] Executor Factory – Test Provider Factory Implementation : Enforce more secure client authentication method when client registration

testsuite/integration-arquillian/servers/auth-server/services/testsuite-providers/src/main/java/org/keycloak/testsuite/services/clientpolicy/executor/TestClientAuthenticationExecutorFactory.java

Client Policy Basics Classes

[56] Executor – Test Provider Implementation : Enforce Proof Key for Code Exchange (PKCE)

testsuite/integration-arquillian/servers/auth-server/services/testsuite-providers/src/main/java/org/keycloak/testsuite/services/clientpolicy/executor/TestPKCEEnforceExecutor.java

[57] Executor Factory – Test Provider Factory Implementation : Enforce Proof Key for Code Exchange (PKCE)

testsuite/integration-arquillian/servers/auth-server/services/testsuite-providers/src/main/java/org/keycloak/testsuite/services/clientpolicy/executor/TestPKCEEnforceExecutorFactory.java

[58] Test Provider Condition Implementation Registration

testsuite/integration-arquillian/servers/auth-server/services/testsuite-providers/src/main/resources/META-INF/services/org.keycloak.services.clientpolicy.condition.ClientPolicyConditionProviderFactory

[59] Test Provider Executor Implementation Registration

testsuite/integration-arquillian/servers/auth-server/services/testsuite-providers/src/main/resources/META-INF/services/org.keycloak.services.clientpolicy.executor.ClientPolicyExecutorProviderFactory

Client Policy Basics Classes

[60] Arquillian Integration Test – Client Policy Basics

testsuite/integration-arquillian/tests/base/src/test/java/org/keycloak/testsuite/client/ClientPolicyBasicsTest.java

[61] Arquillian Integration Test – Client Policy Logger Log Level Control

testsuite/integration-arquillian/tests/base/src/test/resources/log4j.properties

End