For keycloak FAPI-SIG

June 2021

Client Policies Practical Guide

This document describes Client Policies that is officially introduced from keycloak 14.

Client Policies is introduced to support security profiles like FAPI in unified and extensible manner.

The intended reader is the following.

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

Preface

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 client settings.

- Code Maintainability/Extensibility/Availability

  It can improve readability of endpoint classes.

- Backward Compatibility *

  It can realize what the current Client Registration Policies do

* : not yet completely realized.

Usability – auto verify/configure client settings

When client app's developers want to follow some security requirements, they need to consider which values are appropriate for their client settings to follow these requirement one by one securely.

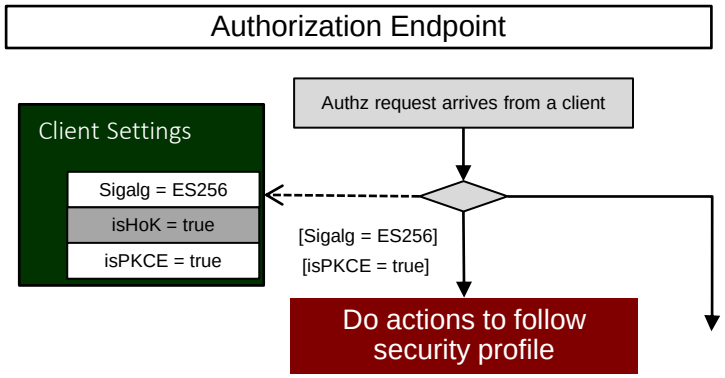
[auto verify]

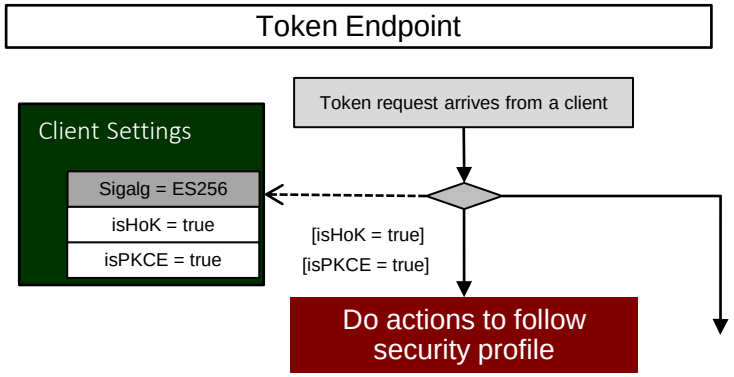
By client policies, security profile related client settings can be automatically verified to meet security requirements. If not, keycloak tells it to client app's developers / keycloak admin so that they can re-set them properly.

[auto configure]

By client policies, security profile related client settings can be automatically configured properly to meet security requirements.

Usability – client setting oriented to policy oriented *





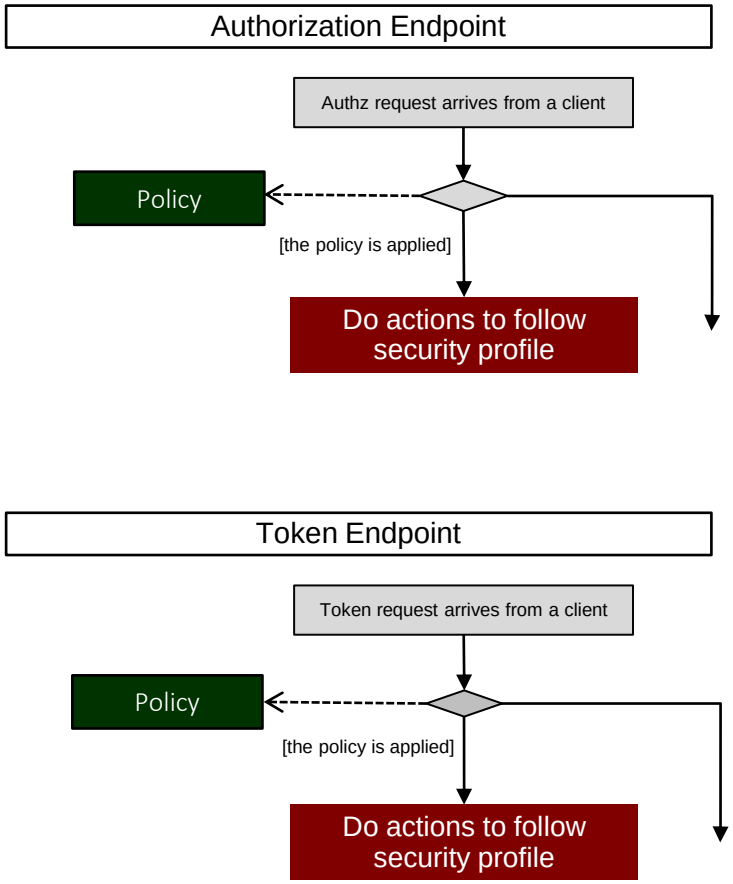
To follow a security profile like FAPI, its related actions need to be executed on a request from a client on each endpoint.

One way for a client to follow a security profile is to setup its configurations that induce keycloak to do such actions to its request on each endpoint.

To do so, a client app developer (or keycloak admin on behalf of them) need to setup and maintain such configurations properly, which cost them.

To accommodate several security profiles and clients, keycloak and keycloak's admin needs to manage a lot of such items, which cost them.

Usability – client setting oriented to policy oriented *



* : not yet completely realized.

The other way for a client to follow a security profile is to introduce the concept “policy” which determines a set of requests from clients and “profiles” which is accompanied with a policy and determines actions executed on a request from a client on each endpoint to follow a security profile.

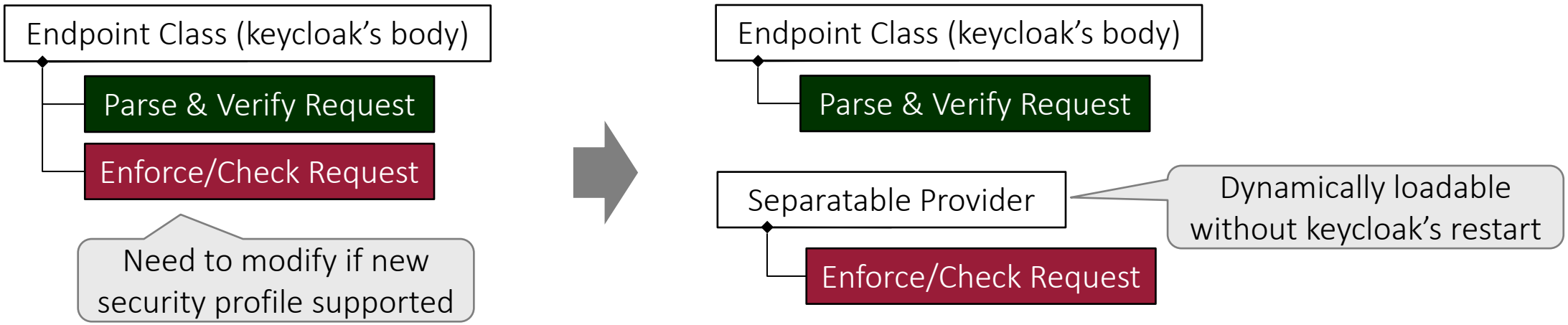
To do so, a client app developer (or keycloak admin on behalf of them) need not to do anything for security profile specific configuration.

To accommodate several security profiles and clients, keycloak and keycloak’s admin need not to manage such items but need to manage “policy” and “profile”, such burden does not increase when the number of clients increases.

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.



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

* : not yet completely realized.

Scope

Scope

This document's target keycloak version is 14.

This document mainly covers the followings :

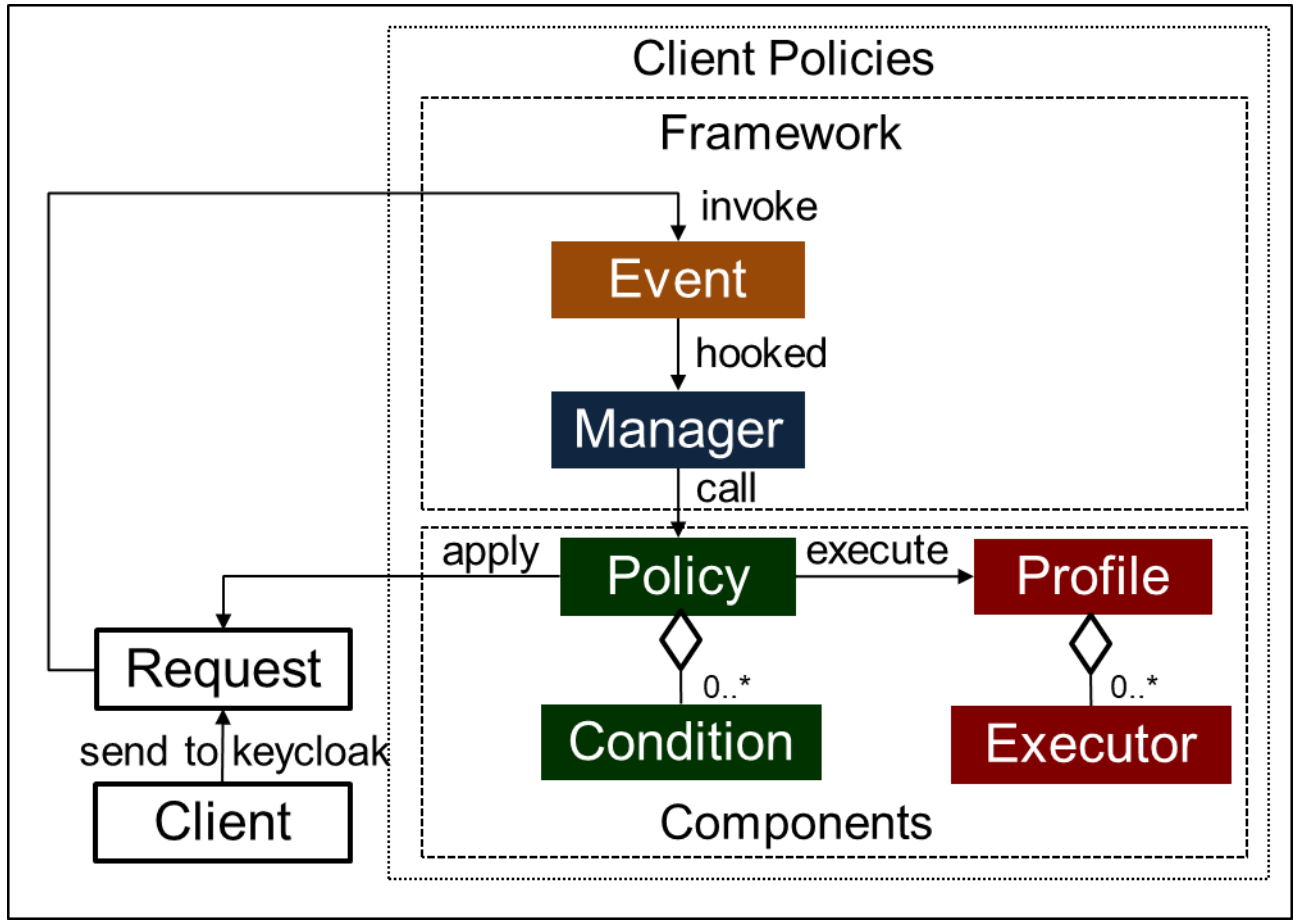
- Design document : Client Policies draft version 2
- JIRA Ticket : KEYCLOAK-14189 Client Policy : Basics
- JIRA Ticket : KEYCLOAK-16805 Client Policy : Support New Admin REST API (Implementation)
- JIRA Ticket : KEYCLOAK-14209 Client policies admin console support
- Pull Request : #7104
- Pull Request : #7780
- Pull Request : #7969

Especially, this document does not cover the followings :

- Other unresolved JIRA sub tickets of KEYCLOAK-13933 Client Policies
  - New Admin Console UI
  - Client Registration Policies Migration

Mechanism

Layout



Basically, Client Policies consists of two parts.

- Framework

  Load & run components.

  Implemented on keycloak's body.

- Components

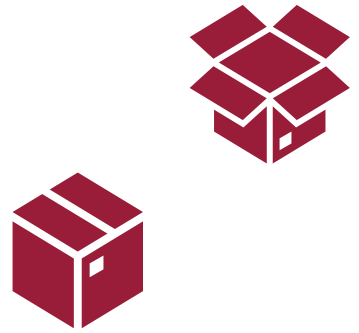
  Determine which security profile related actions are executed on which client's request to keycloak.

Elements

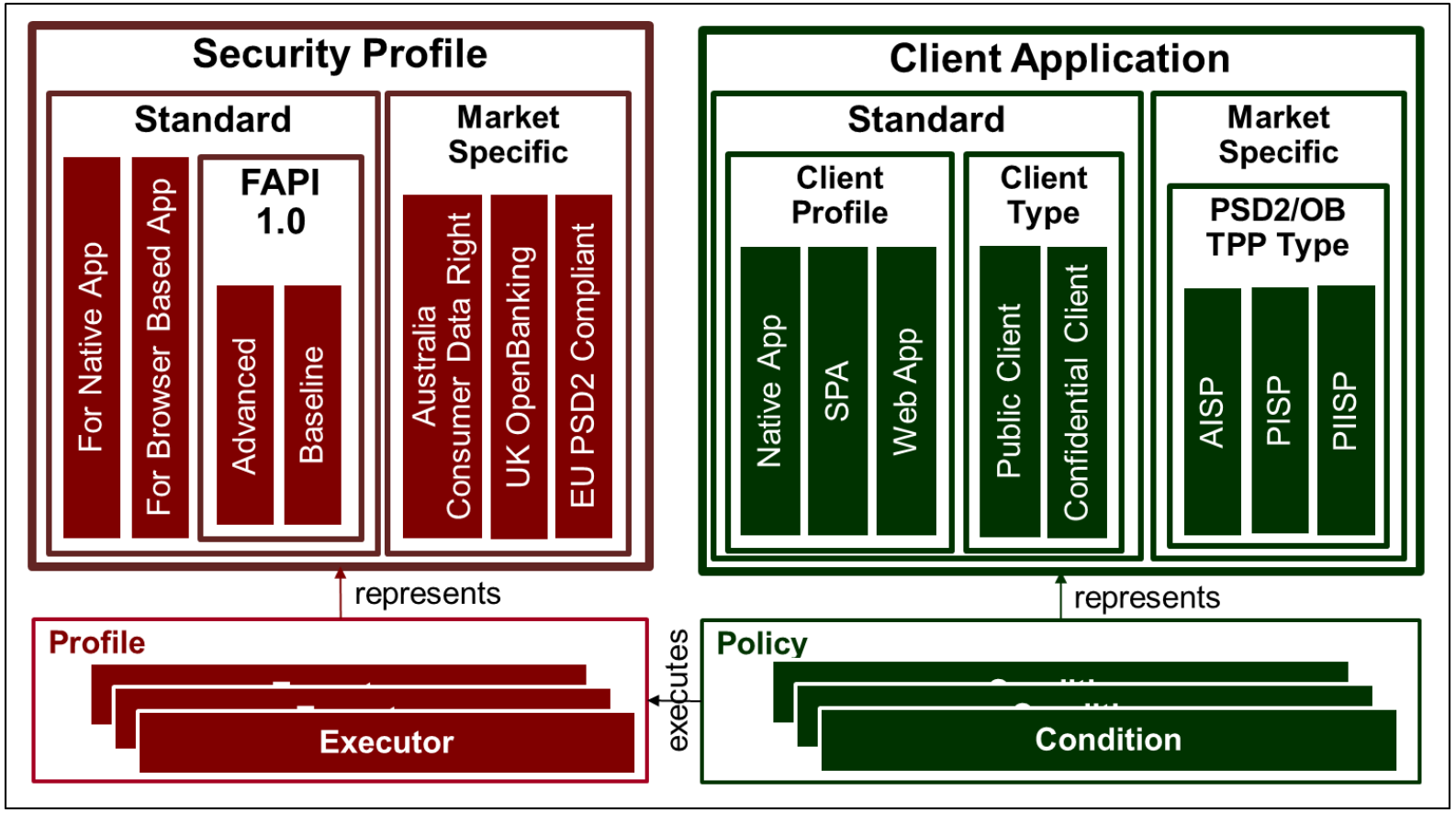
[Components]
- Executor
- Profile
- Condition
- Policy

[Framework]
- Event
- Context
- Manager

[Misc]
- Vote
- Exception
- Test



Components



- Executor : what to do

  Implement security profile related action.

- Profile : what to do as a security profile

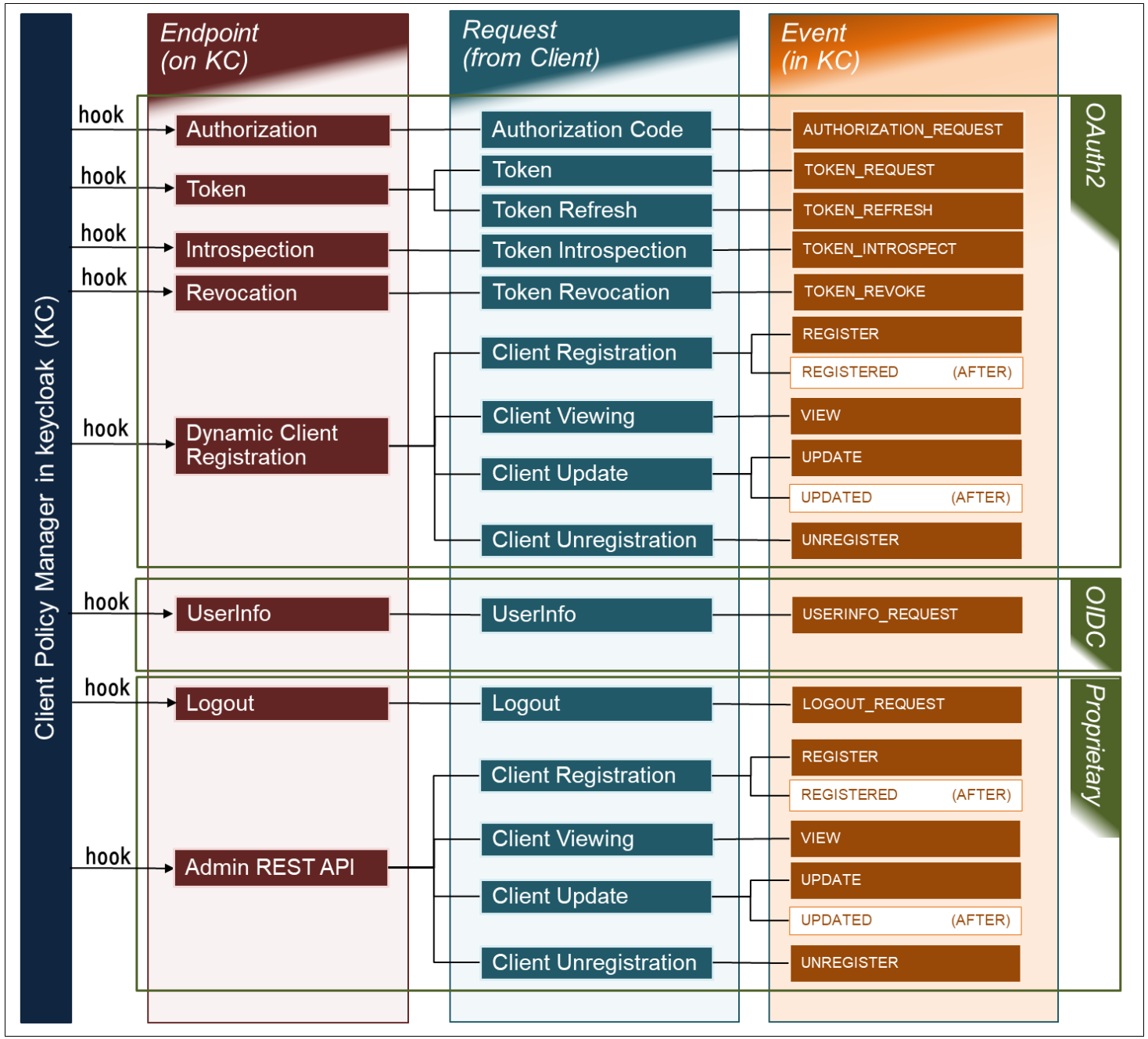
  Bundle of executors to follow a security profile.

- Condition : to which client's request

  Determine to which clients' requests profiles are executed.

- Policy : to which client's request for executing profiles

  Bundle of conditions for defining to which clients' requests profiles are executed.



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 policies to determine whether accompanied profiles executed. If so, execute profiles.

Misc

- Vote

  The return value of evaluating condition

  “YES” : the client satisfies this condition

  “NO” : the client does not satisfy this condition

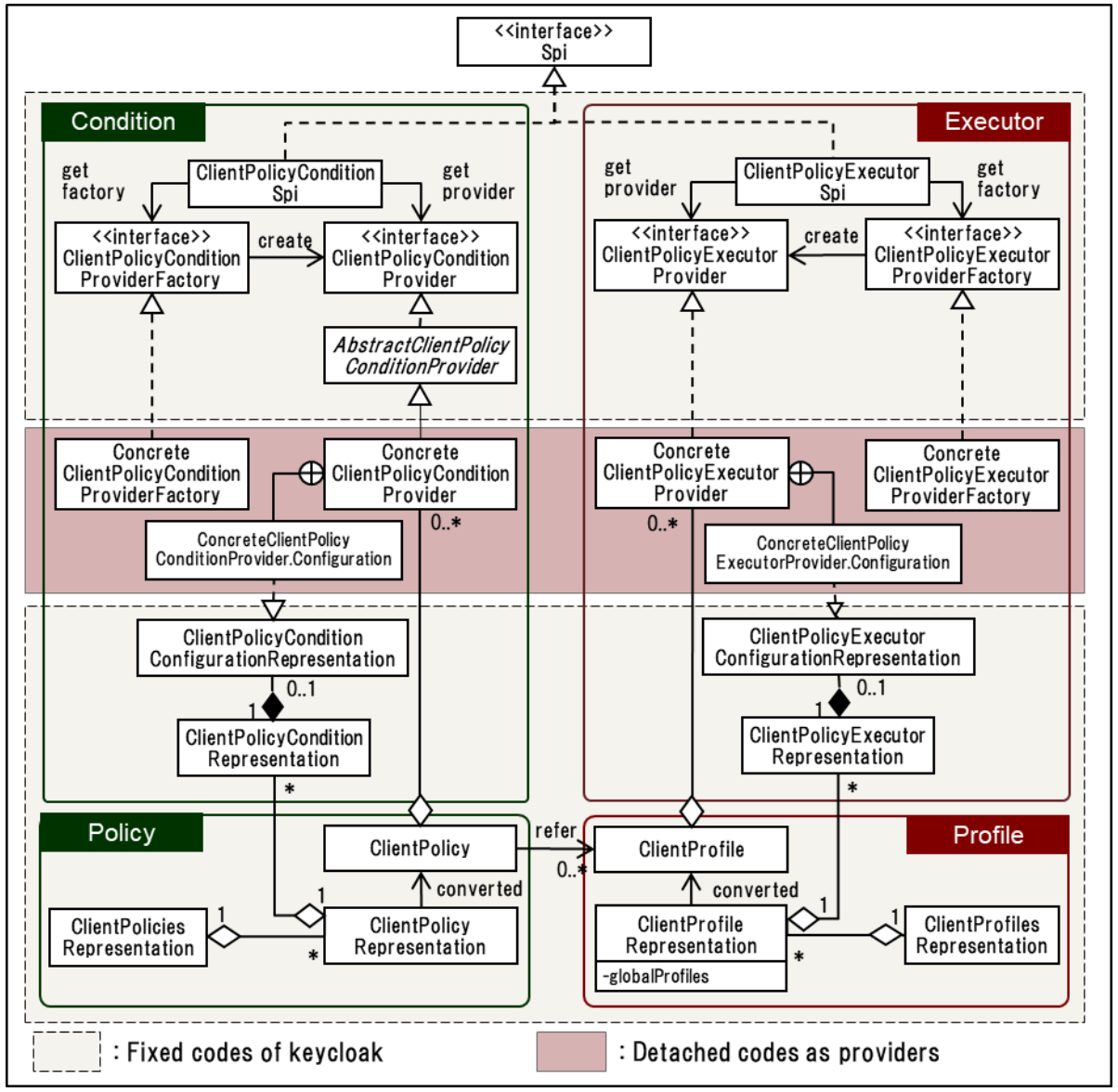
  “ABSTAIN” : skipping the evaluation of this condition
- Exception

  Showing errors occurred when evaluating condition and executing executor.
- Test

  Testing Client Policies feature by Arquillian Integration Test framework.

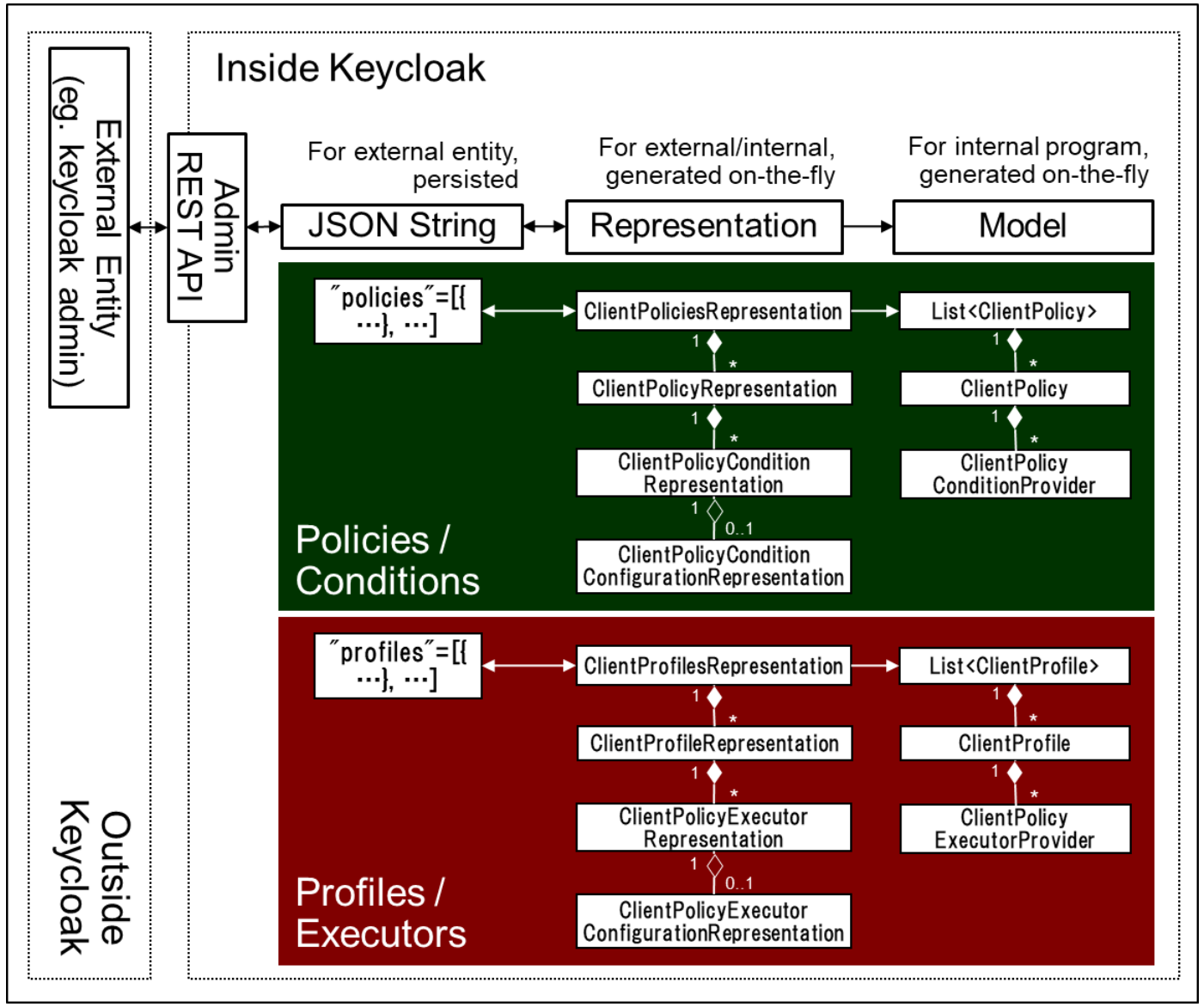
Internals

Class Layout



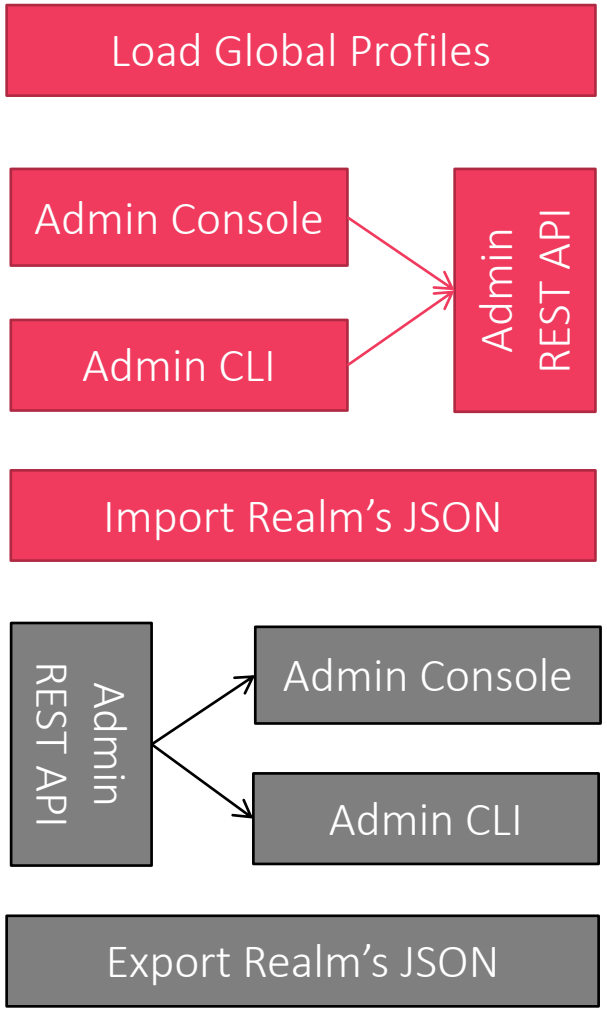
- Fixed codes of keycloak
  - Client policy basics (Framework)
  - Interface of condition/executor
  - Representations of policy/profile
- Detached codes as providers
  - Concrete condition/executor

Representation



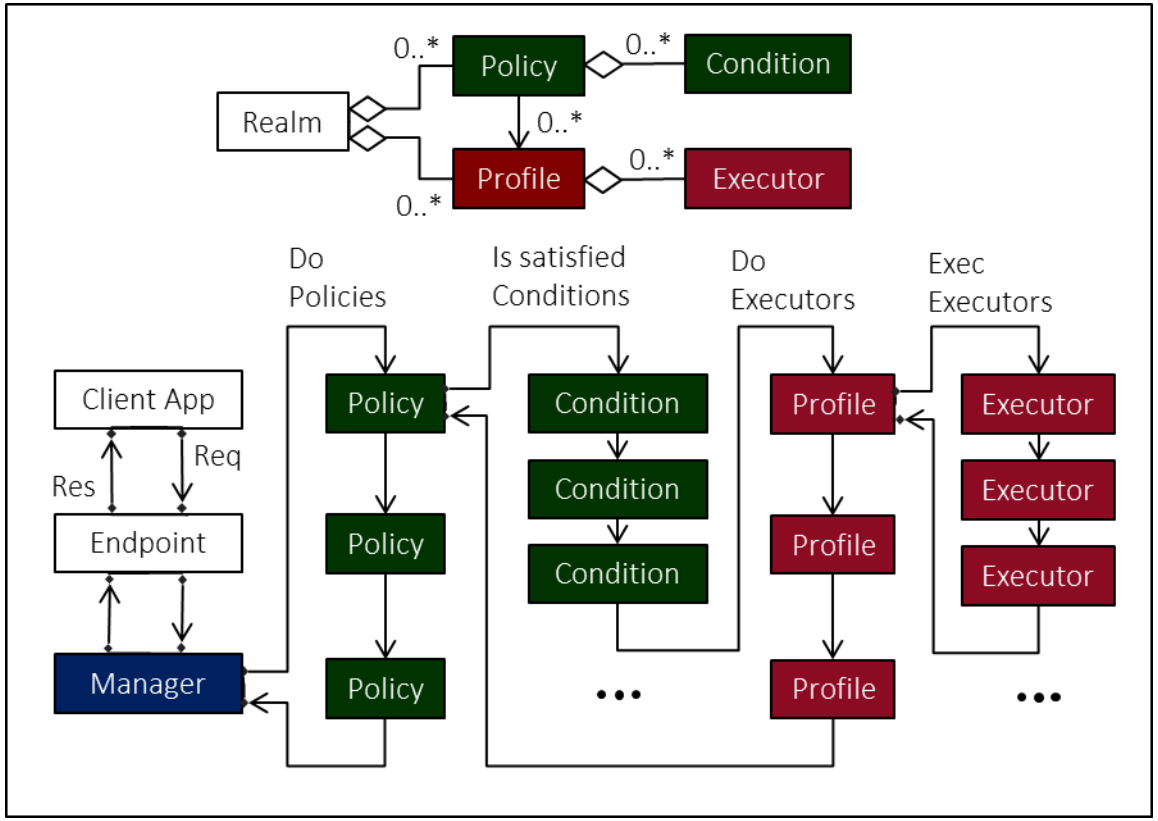
- External Representation
  - JSON String
  - Persisted as it is
- Interim Representation
  - "Representation" POJO
  - Bridges external and internal representation
- Internal Representation
  - "Model" logic implemented
  - Provides logics for policy/profile

Representation : Input / Output



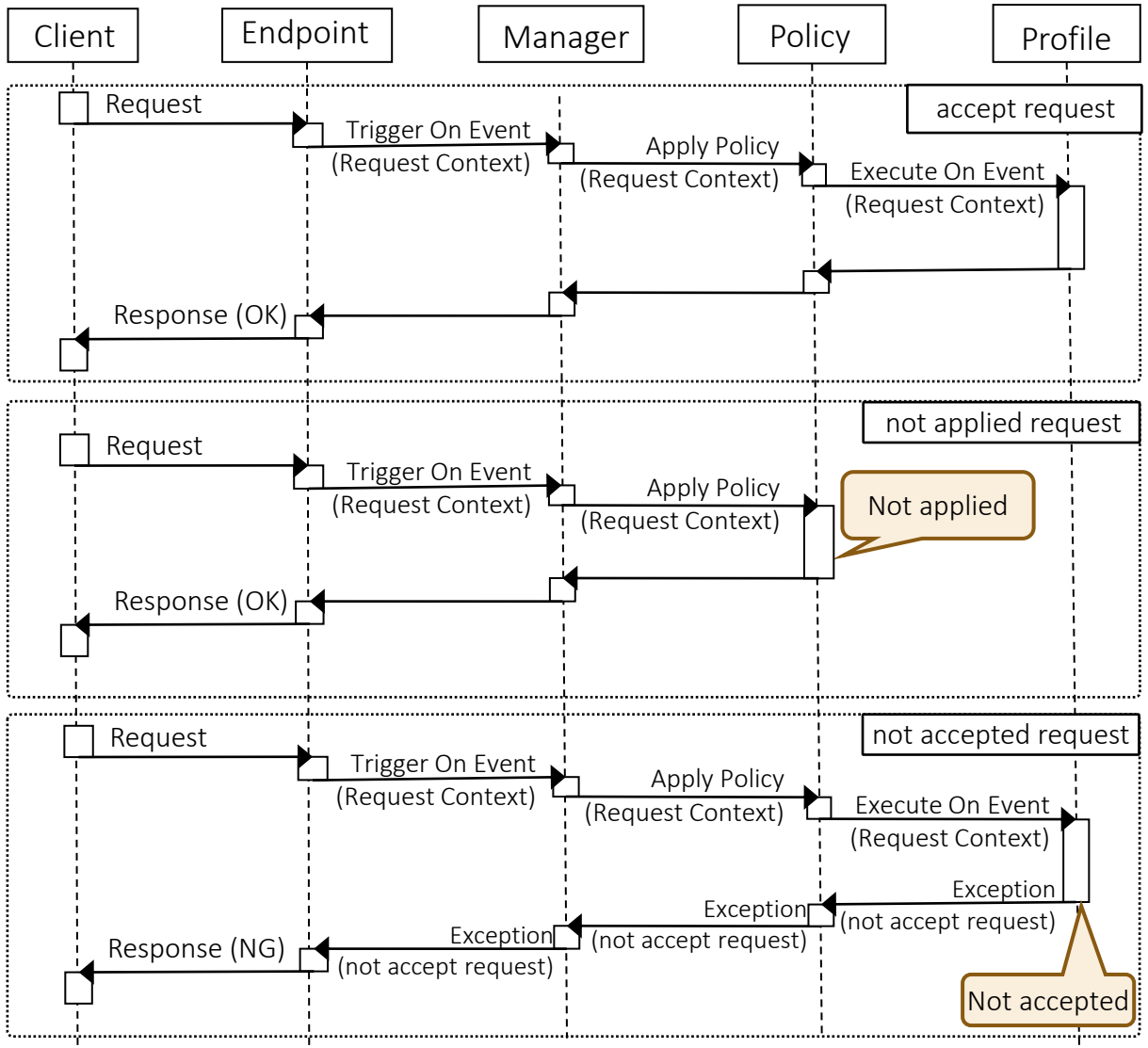
- load global profiles from keycloak's binary when keycloak is booted. Keycloak's admin cannot interfere it.
- put policies/profiles via Admin Console/CLI which translate its own representation to Admin REST API's one.
- put policies/profiles as part of realm's JSON representation.
- get policies/profiles via Admin Console/CLI which translate Admin REST API's one to its own representation.
- get policies/profiles as part of realm's JSON representation which does not include global profiles.

Cardinality, Duplication, Order of Execution



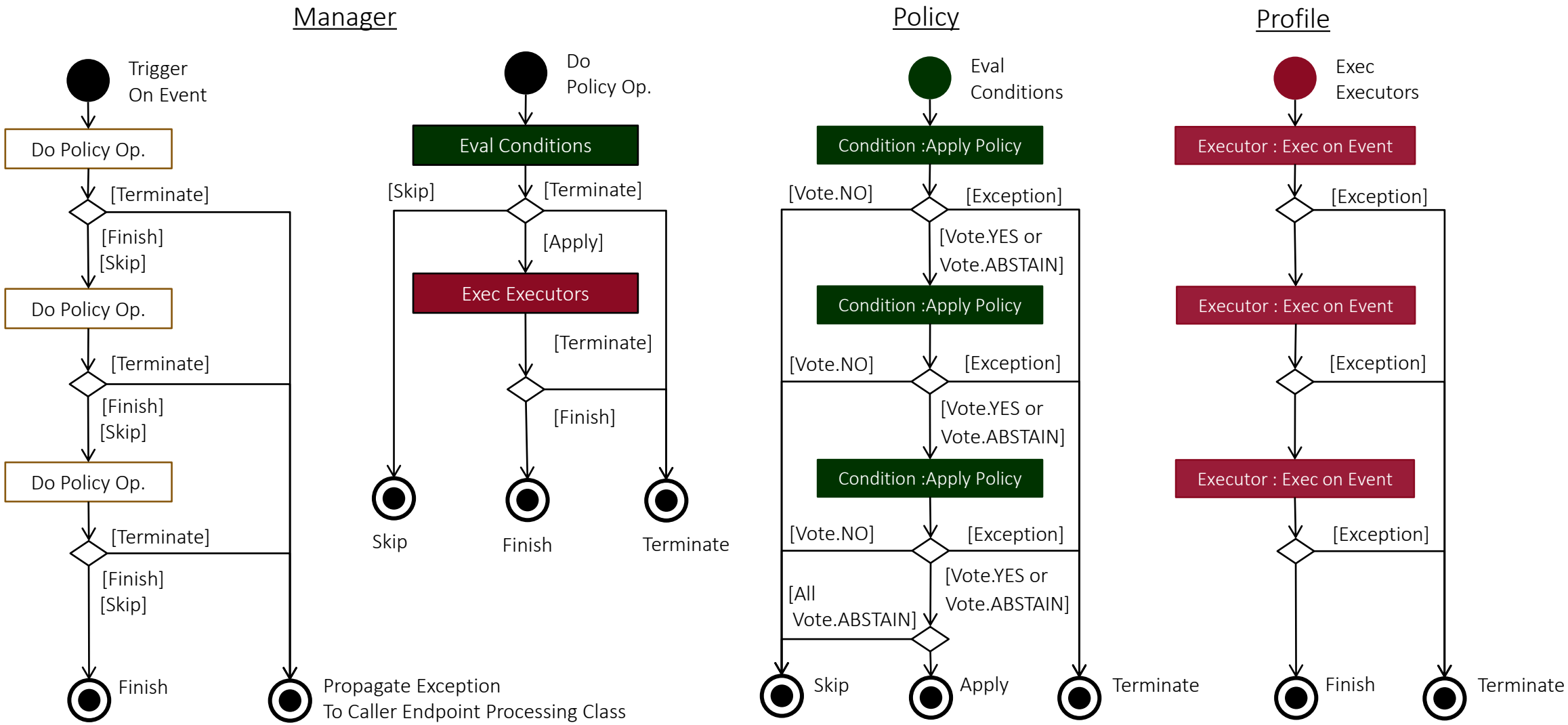
- [Cardinality] Realm can accommodate multiple policies and profiles. The number of policies and profiles is unlimited.
- [Cardinality] Policy can accommodate multiple conditions and profile can accommodate multiple executors. The number of conditions/executors are unlimited.
- [Duplication] It is allowed to duplicate conditions and executors whose providers are the same.
- [Order of Execution] We can not specify the order of execution of policies/profiles/conditions/executors.

Client Policies Evaluation : Flow Diagram

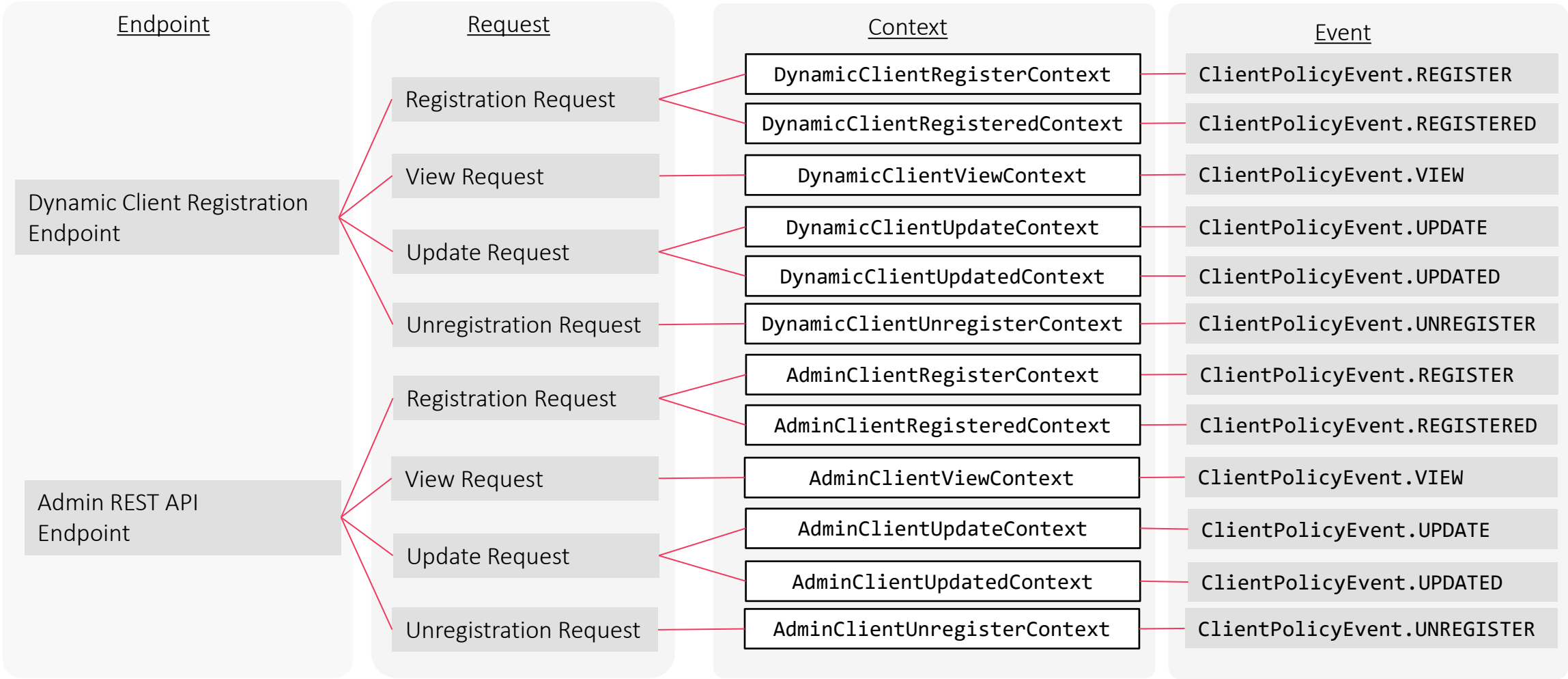


- On receiving a request from a client, endpoint class asks manager to do client policies operations with passing this request's context data.
- Manager evaluate all policies and execute profiles if applied.
- [Error Notification] If some executor determines that the client's request is not acceptable, throw Client Policy Exception to notify caller endpoint class of it. Endpoint class catch it and crafts an error response to the client.
- [Error Notification] Exception is thrown when something wrong happens on evaluating policies and executing profiles.

Client Policies Evaluation : Activity Diagram

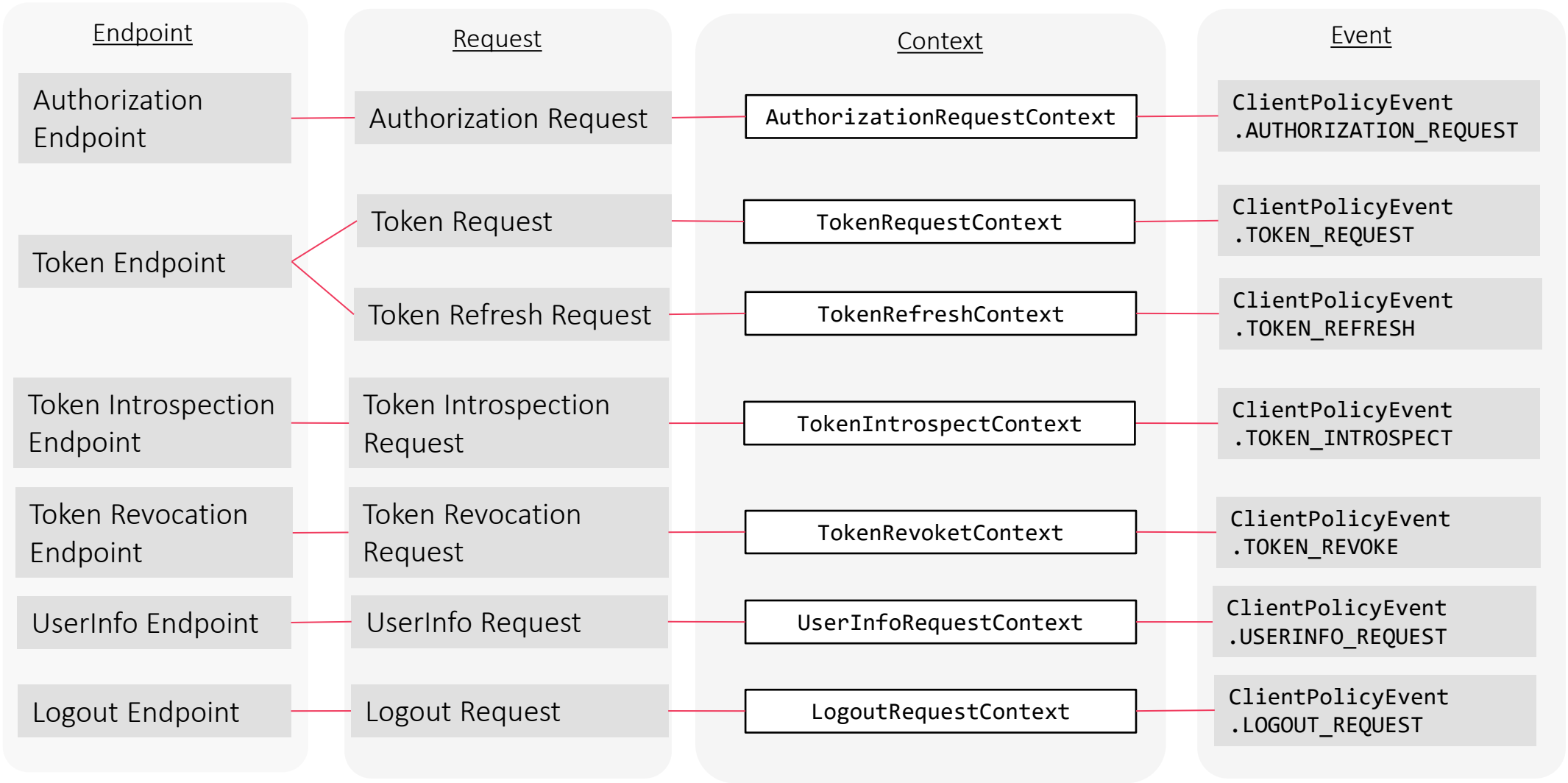


Event & Context



A request received on an endpoint is expressed as a context that is used when evaluating policy/condition and executing profile/execution.

Event & Context



Profile type : global vs normal

Global profile is presented by keycloak as built-in.

Keycloak’s admin cannot create their own global profiles nor edit existing global profiles.

It aim is to provide ways of securing client applications out of the box. keycloak’s admin needs not define such profiles by themselves.

	Normal	Global
Scope	Per realm	All over realms
Editable by keycloak’s admin	Yes	No
Export as realm’s JSON	Yes	No

JSON Representation : Executor

Executor

```
{
        "executor": "secure-client-authenticator",
        "configuration": {
                "allowed-client-authenticators": [
                        "client-jwt",
                        "client-x509"
                ],
                "default-client-authenticator": "client-jwt"
        }
},
{
        "executor": "secure-client-uris",
        "configuration": {}
}
```

Executor Provider's ID

Executor's configuration (depending on each executor)

Need "configuration" even if an executor does not have any configuration.

JSON Representation : Profile

Profile

```
{
        "name": "fapi-1-baseline",
        "description": "Client profile, which enforce clients to conform
                        'Financial-grade API Security Profile 1.0 - Part 1: Baseline' specification.",
        "executors": [
                {
                        "executor": "secure-session",
                        "configuration": {}
                },
          …
        ]
},
{
         "name": “no-executor-profile",
         "description": “sample profile",
         "executors": []
}
```

“name” must be unique in the same profiles.

Executor

The profile without any executor is allowed.

JSON Representation : Profiles

Profiles

```
{
        "profiles": [
                {
                        "name": "fapi-1-baseline",
                        "description": "enforce clients to conform FAPI 1.0 Baseline.",
                        "executors": [
                                {
                                        "executor": "secure-session",
                                        "configuration": {}
                                },
                ...
        ]
}
```

Only one “profiles” exists on a realm

Profile

JSON Representation : Condition

Condition

```
{
        "condition":"client-updater-context",
        "configuration": {
                "update-client-source": [
                        "ByAuthenticatedUser",
                        "ByInitialAccessToken",
                        "ByRegistrationAccessToken“
                ]
        }
},
{
        "condition":"any-client",
        "configuration":{}
}
```

Condition Provider’s ID

Condition’s configuration (depending on each condition)

Need “configuration” even if a condition does not have any configuration.

JSON Representation : Policy

Policy

```
{
        "name": "MyPolicy",
        "description": "Porishii Sono Ichi",
        "enabled": true,
        "conditions": [
                {
                        "condition": "client-roles",
                        "configuration": {
                                "roles": [ "sample-client-role“ ]
                        }
                },
                …
        ]
}, {
        "name": “sample-policy",
        "description": “Sample Policy",
        "enabled": false,
        "conditions": []
}
```

“name” must be unique in the same policies.

Condition can be enabled/disabled.

Condition

The policy without any condition is allowed.

JSON Representation : Policies

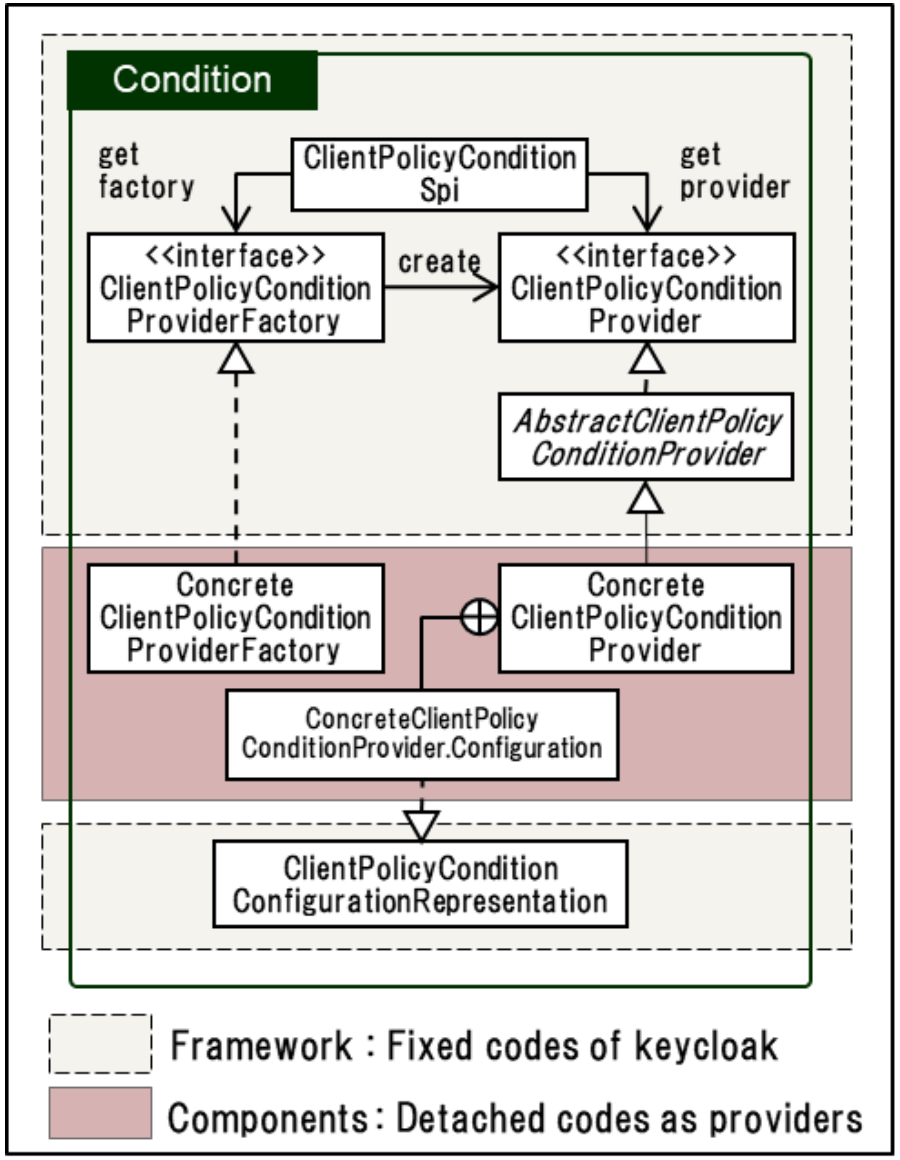
Policies

```
{
        “policies”: [
                {
                        "name": "MyPolicy",
                        "description": "Porishii Sono Ichi",
                        "enabled": true,
                        "conditions": [
                                {
                                        "condition": "client-roles",
                                        "configuration": {
                                                "roles": [ "sample-client-role“ ]
                                        }
                                },
                        …
                },
            …
        ]
},
```

Only one “policies” exists on a realm

Profile

Condition : How to Implement



- There are two types of conditions
  - No-config condition : condition without config
  - Configurable condition : condition with config
- Developers need to implement their own custom
  - `ClientPolicyConditionProvider`
  - `ClientPolicyConditionProviderFactory`
- Developers of configurable condition need to implement their own custom
  - `ClientPolicyConditionProvider.Configuration`

as an inner class of `ClientPolicyConditionProvider`.

No-config condition : How to Implement provider

1. Extend
`AbstractClientPolicyConditionProvider<ClientPolicyConditionConfigurationRepresentation>`

```
public class AnyClientCondition extends AbstractClientPolicyConditionProvider<ClientPolicyConditionConfigurationRepresentation> {

    public AnyClientCondition(KeycloakSession session) {
        super(session);
    }
…
```

This condition does not have its configuration so that set
`ClientPolicyConditionConfigurationRepresentation` itself as generic type.

2. Override `getConditionConfigurationClass`

```
…
    @Override
    public Class<ClientPolicyConditionConfigurationRepresentation> getConditionConfigurationClass() {
        return ClientPolicyConditionConfigurationRepresentation.class;
    }
…
```

This condition does not have its configuration so that return
`ClientPolicyConditionConfigurationRepresentation` itself.

No-config condition : How to Implement provider

3. Override `getProviderId`

```
…
    @Override
    public String getProviderId() {
        return AnyClientConditionFactory.PROVIDER_ID;
    }
…
```

By convention, its provider factory define this provider ID and the provider refers to it here.

4. Override `applyPolicy`

```
…
    @Override
    public ClientPolicyVote applyPolicy(ClientPolicyContext context) throws ClientPolicyException {
        return ClientPolicyVote.YES;
    }
}
```

Implement codes on each Event.

Dependent on Event, `ClientPolicyContext` argument is casted into the corresponding Context class.

Return `ABSTAIN` if the Event is not evaluated

No-config condition : How to Implement provider factory

1. Implement `ClientPolicyConditionProviderFactory`

```
public class AnyClientConditionFactory implements ClientPolicyConditionProviderFactory {
…
```

2. Define Provider ID

```
…
    public static final String PROVIDER_ID = "any-client";
…
```

By convention, define PROVIDER_ID showing Provider ID that identifies the provide so that it must be unique.

3. Override `getId`

```
…
    @Override
    public String getId() {
        return PROVIDER_ID;
    }
…
```

By convention, return PROVIDER_ID.

No-config condition : How to Implement provider factory

4. Override `getConfigProperties`

```
…
    @Override
    public List<ProviderConfigProperty> getConfigProperties() {
        return Collections.emptyList();
    }
}
```

This method returns initial values of configurations. There is not configuration so that returns empty list.

Configurable condition : How to Implement provider

1. Extend `AbstractClientPolicyConditionProvider<`"Your Custom Provider"`.Configuration>`

```
public class ClientAccessTypeCondition extends AbstractClientPolicyConditionProvider<ClientAccessTypeCondition.Configuration> {
…
```

2. Define this custom provider's configuration.

```
…
    public static class Configuration extends ClientPolicyConditionConfigurationRepresentation {

        protected List<String> type;

        public List<String> getType() {
            return type;
        }

        public void setType(List<String> type) {
            this.type = type;
        }
    }
…
```

This condition have its configuration and it must be defined as an inner class and needs to extend `ClientPolicyConditionConfigurationRepresentation` .

Configurable condition : How to Implement provider

3. Override `getConditionConfigurationClass`

```
…
    @Override
    public Class<Configuration> getConditionConfigurationClass() {
        return Configuration.class;
    }
…
```

Return the class defined in step 2.

4. Override `getProviderId`

```
…
    @Override
    public String getProviderId() {
        return ClientAccessTypeConditionFactory.PROVIDER_ID;
    }
…
```

By convention, its provider factory define this provider ID and the provider refers to it here.

Configurable condition : How to Implement provider

5. Override `applyPolicy`

```
…
    @Override
    public ClientPolicyVote applyPolicy(ClientPolicyContext context) throws ClientPolicyException {
        switch (context.getEvent()) {
            case AUTHORIZATION_REQUEST:
            case TOKEN_REQUEST:
            case TOKEN_REFRESH:
            case TOKEN_REVOKE:
            case TOKEN_INTROSPECT:
            case USERINFO_REQUEST:
            case LOGOUT_REQUEST:
                if (isClientAccessTypeMatched()) return ClientPolicyVote.YES;
                return ClientPolicyVote.NO;
            default:
                return ClientPolicyVote.ABSTAIN;
        }
    }
…
```

Implement codes on each Event.

Dependent on Event, `ClientPolicyContext` argument is casted into the corresponding Context class.

Return `ABSTAIN` if the Event is not evaluated

Configurable condition : How to Implement provider factory

1. Implement `ClientPolicyConditionProviderFactory`

```
public class ClientAccessTypeConditionFactory implements ClientPolicyConditionProviderFactory {
…
```

2. Define Provider ID

```
…
    public static final String PROVIDER_ID = "client-access-type";
…
```

By convention, define PROVIDER_ID showing Provider ID that identifies the provide so that it must be unique.

3. Override `getId`

```
…
    @Override
    public String getId() {
        return PROVIDER_ID;
    }
…
```

By convention, return PROVIDER_ID.

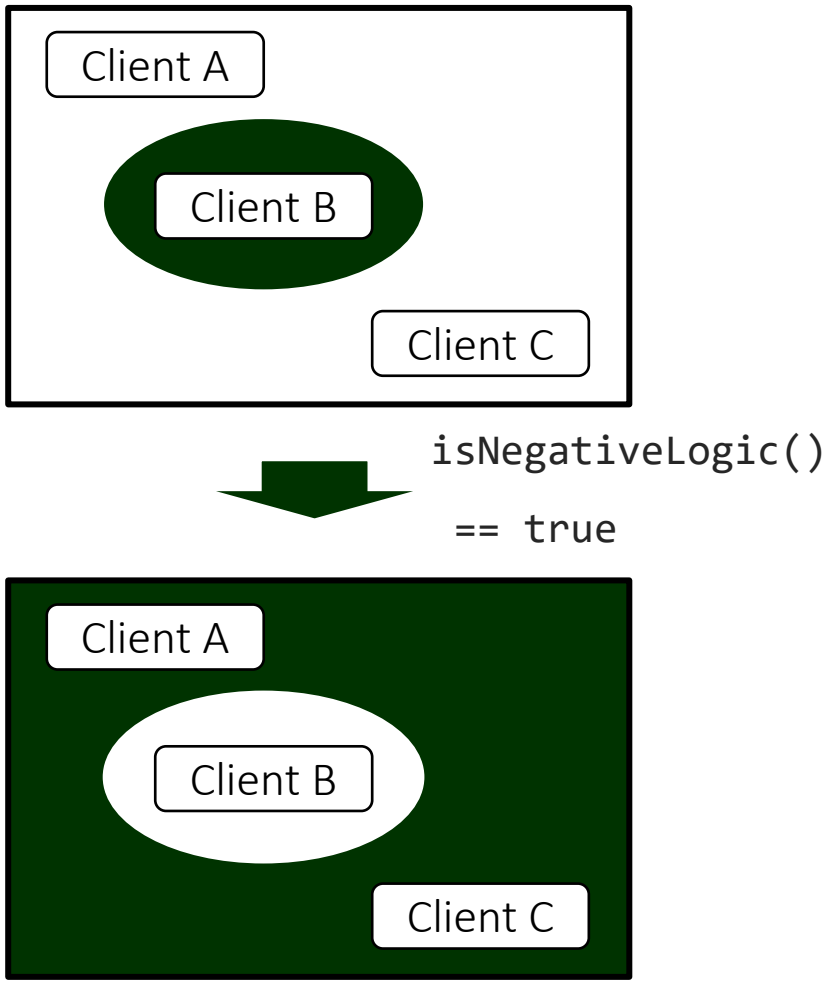
Configurable condition : How to Implement provider factory

4. Override `getConfigProperties`

```
…
    private static final List<ProviderConfigProperty> configProperties = new ArrayList<ProviderConfigProperty>();

    static {
        ProviderConfigProperty property;
        property = new ProviderConfigProperty(TYPE, "client-accesstype.label", "client-accesstype.tooltip",
                                              ProviderConfigProperty.MULTIVALUED_LIST_TYPE, TYPE_CONFIDENTIAL);
        List<String> updateProfileValues = Arrays.asList(TYPE_CONFIDENTIAL, TYPE_PUBLIC, TYPE_BEARERONLY);
        property.setOptions(updateProfileValues);
        configProperties.add(property);
    }

    @Override
    public List<ProviderConfigProperty> getConfigProperties() {
        return configProperties;
    }
}
```

This method returns initial values of configurations.

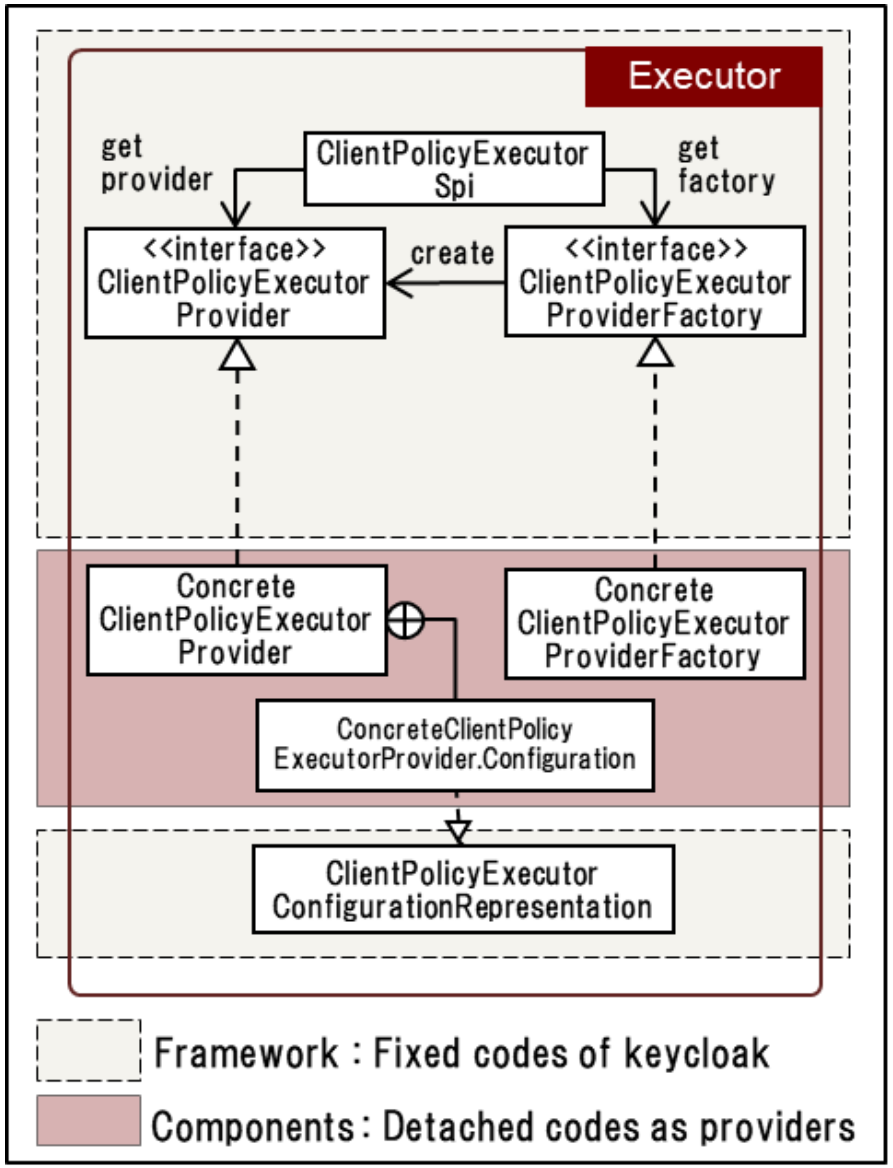
Condition : Negative Logic



By using `"is-negative-logic"` configuration item that can be used as default, it can revert to which client's request is satisfied with a condition.

```
public class ClientPolicyConditionConfigurationRepresentation {

    private Map<String, Object> configAsMap = new HashMap<>();

    @JsonProperty("is-negative-logic")
    private Boolean negativeLogic;

    public Boolean isNegativeLogic() {
        return negativeLogic;
    }
…
```

Executor : How to Implement



- There are two types of executors
  - No-config executor : executor without config
  - Configurable executor : executor with config
- Developers need to implement their own custom
  - `ClientPolicyExecutorProvider`
  - `ClientPolicyExecutorProviderFactory`
- Developers of configurable executor need to implement their own custom
  - `ClientPolicyExecutorProvider.Configuration`

as an inner class of `ClientPolicyExecutorProvider`.

No-config executor : How to Implement provider

1. Implement `ClientPolicyExecutorProvider<ClientPolicyExecutorConfigurationRepresentation>`

```
public class ConfidentialClientAcceptExecutor implements ClientPolicyExecutorProvider<ClientPolicyExecutorConfigurationRepresentation> {
    protected final KeycloakSession session;

    public ConfidentialClientAcceptExecutor(KeycloakSession session) {
        this.session = session;
    }
…
```

This executor does not have its configuration so that set `ClientPolicyExecutorConfigurationRepresentation` itself as generic type.

2. Override `getProviderId`

```
…
    @Override
    public String getProviderId() {
        return ConfidentialClientAcceptExecutorFactory.PROVIDER_ID;
    }
…
```

By convention, its provider factory define this provider ID and the provider refers to it here.

No-config executor : How to Implement provider

4. Override `executeOnEvent`

```
…
    @Override
    public void executeOnEvent(ClientPolicyContext context) throws ClientPolicyException {
        switch (context.getEvent()) {
            case AUTHORIZATION_REQUEST:
            case TOKEN_REQUEST:
            checkIsConfidentialClient();
                return;
            default:
                return;
        }
    }
}
```

Implement codes on each Event.

Dependent on Event, `ClientPolicyContext` argument is casted into the corresponding Context class .

Throw `ClientPolicyException` the client's request is prohibited due to not complying with the security profile implemented by the executor.

No-config executor : How to Implement provider factory

1. Implement `ClientPolicyExecutorProviderFactory`

```
public class ConfidentialClientAcceptExecutorFactory implements ClientPolicyExecutorProviderFactory {
…
```

2. Define Provider ID

```
…
    public static final String PROVIDER_ID = "confidential-client";
…
```

By convention, define PROVIDER_ID showing Provider ID that identifies the provide so that it must be unique.

3. Override `getId`

```
…
    @Override
    public String getId() {
        return PROVIDER_ID;
    }
…
```

By convention, return PROVIDER_ID.

No-config executor : How to Implement provider factory

4. Override `getConfigProperties`

```
…
    @Override
    public List<ProviderConfigProperty> getConfigProperties() {
        return Collections.emptyList();
    }
}
```

This method returns initial values of configurations. There is not configuration so that returns empty list.

Configurable executor : How to Implement provider

1. Implement `ExtendClientPolicyExecutorProvider<`"Your Custom Provider"`.Configuration>`

```
public class SecureClientAuthenticatorExecutor implements ClientPolicyExecutorProvider<SecureClientAuthenticatorExecutor.Configuration> {
…
```

2. Define this custom provider's configuration.

```
…
 public static class Configuration extends ClientPolicyExecutorConfigurationRepresentation {
        @JsonProperty("allowed-client-authenticators")
        protected List<String> allowedClientAuthenticators;
        @JsonProperty("default-client-authenticator")
        protected String defaultClientAuthenticator;

        public List<String> getAllowedClientAuthenticators() {
            return allowedClientAuthenticators;
        }

        public void setAllowedClientAuthenticators(List<String> allowedClientAuthenticators) {
            this.allowedClientAuthenticators = allowedClientAuthenticators;
        }
…
```

This executor have its configuration and it must be defined as an inner class and needs to extend `ClientPolicyExecutorConfigurationRepresentation` .

Configurable executor : How to Implement provider

3. Override `getExecutorConfigurationClass`

```
@Override
public Class<Configuration> getExecutorConfigurationClass() {
    return Configuration.class;
}
```

Return the class defined in step 2.

4. Override `setupConfiguration`

```
@Override
public void setupConfiguration(SecureClientAuthenticatorExecutor.Configuration config) {
    this.configuration = config;
}
```

Set configuration to this executor.

5 . Override `getProviderId`

```
@Override
public String getProviderId() {
    return SecureClientAuthenticatorExecutorFactory.PROVIDER_ID;
}
```

By convention, its provider factory define this provider ID and the provider refers to it here.

Configurable executor : How to Implement provider

6. Override `executeOnEvent`

```
@Override
   public void executeOnEvent(ClientPolicyContext context) throws ClientPolicyException {
       switch (context.getEvent()) {
       case REGISTER:
       case UPDATE:
           ClientCRUDContext clientUpdateContext = (ClientCRUDContext)context;
           autoConfigure(clientUpdateContext.getProposedClientRepresentation());
           validateDuringClientCRUD(clientUpdateContext.getProposedClientRepresentation());
           break;
       case TOKEN_REQUEST:
       case TOKEN_REFRESH:
       case TOKEN_REVOKE:
       case TOKEN_INTROSPECT:
       case LOGOUT_REQUEST:
           validateDuringClientRequest();
       default:
           return;
       }
   }
```

Implement codes on each Event.

Dependent on Event, `ClientPolicyContext` argument is casted into the corresponding Context class .

Throw `ClientPolicyException` the client's request is prohibited due to not complying with the security profile implemented by the executor.

Configurable executor : How to Implement provider factory

1. Implement `ClientPolicyExecutorProviderFactory`

```
public class SecureClientAuthenticatorExecutorFactory implements ClientPolicyExecutorProviderFactory {
…
```

2. Define Provider ID

```
…
 public static final String PROVIDER_ID = "secure-client-authenticator";
…
```

By convention, define PROVIDER_ID showing Provider ID that identifies the provide so that it must be unique.

3. Override `getId`

```
…
    @Override
    public String getId() {
        return PROVIDER_ID;
    }
…
```

By convention, return PROVIDER_ID.

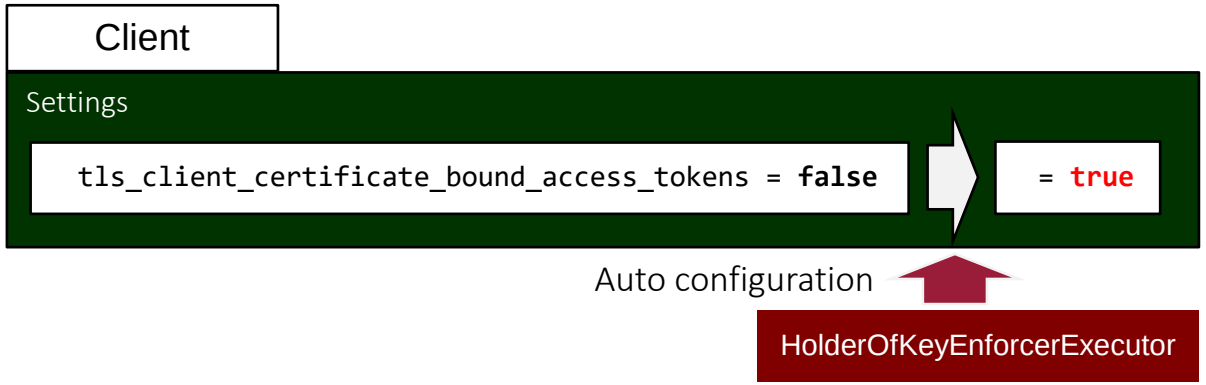
Configurable condition : How to Implement provider factory

4. Override `getConfigProperties`

```
    private List<ProviderConfigProperty> configProperties = new ArrayList<>();

    @Override
    public void postInit(KeycloakSessionFactory factory) {
        List<String> clientAuthProviders = factory.getProviderFactoriesStream(ClientAuthenticator.class)
                .map(ProviderFactory::getId)
                .collect(Collectors.toList());

        ProviderConfigProperty allowedClientAuthenticatorsProperty = new ProviderConfigProperty(
                ALLOWED_CLIENT_AUTHENTICATORS, "Allowed Client Authenticators", "",
                ProviderConfigProperty.MULTIVALUED_LIST_TYPE, null);
        allowedClientAuthenticatorsProperty.setOptions(clientAuthProviders);

        ProviderConfigProperty autoConfiguredClientAuthenticator = new ProviderConfigProperty(
                DEFAULT_CLIENT_AUTHENTICATOR, "Default Client Authenticator", "",
                ProviderConfigProperty.LIST_TYPE, JWTClientAuthenticator.PROVIDER_ID);
        autoConfiguredClientAuthenticator.setOptions(clientAuthProviders);

        configProperties = Arrays.asList(allowedClientAuthenticatorsProperty, autoConfiguredClientAuthenticator);
    }

    @Override
    public List<ProviderConfigProperty> getConfigProperties() {
        return configProperties;
    }
```

This method returns initial values of configurations.

Executor : Auto configuration



Executor can set a value onto a client setting item forcibly.

It can prevent client app developers to set their configuration that does not satisfy some security requirements keycloak' admin want to enforce.

```
public static class Configuration extends ClientPolicyExecutorConfigurationRepresentation {
    @JsonProperty("auto-configure")
    protected Boolean autoConfigure;
…
@Override
public void executeOnEvent(ClientPolicyContext context) throws ClientPolicyException {
    HttpRequest request = session.getContext().getContextObject(HttpRequest.class);
    switch (context.getEvent()) {
        case REGISTER:
        case UPDATE:
            ClientCRUDContext clientUpdateContext = (ClientCRUDContext)context;
            autoConfigure(clientUpdateContext.getProposedClientRepresentation());
…
private void autoConfigure(ClientRepresentation rep) {
    if (configuration.isAutoConfigure()) {
        OIDCAdvancedConfigWrapper.fromClientRepresentation(rep).setUseMtlsHoKToken(true);
    }
}
```

[Implementation Notice]

While "negative logic" of condition has been implemented as the framework, "auto configuration" of executor has not. Therefore, developers need to implement this feature by themselves.

Condition/Executor : How to register/unregister to keycloak

Please refer to keycloak's server developer guide :

https://www.keycloak.org/docs/latest/server_development/index.html#registering-provider-implementations

For custom condition/executor implementer, no need to implement custom SPI so that keycloak deployer can used

Admin Console UI

Please refer to concept design of client policies for new admin console:

https://marvelapp.com/prototype/6e70eh2/screen/76353681

It is for new admin console but the existing admin console generally following this design.

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

- [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 classes are used :

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

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

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

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

Tests

- [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
```

Client Policies 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] Representation : Policies

core/src/main/java/org/keycloak/representations/idm/ClientPoliciesRepresentation.java

[4] Representation : Condition Configuration

core/src/main/java/org/keycloak/representations/idm/ClientPolicyConditionConfigurationRepresentation.java

[5] Representation : Condition

core/src/main/java/org/keycloak/representations/idm/ClientPolicyConditionRepresentation.java

[6] Representation : Executor Configuration

core/src/main/java/org/keycloak/representations/idm/ClientPolicyExecutorConfigurationRepresentation.java

[7] Representation : Executor

core/src/main/java/org/keycloak/representations/idm/ClientPolicyExecutorRepresentation.java

Client Policies Classes

[8] Representation : Policy

core/src/main/java/org/keycloak/representations/idm/ClientPolicyRepresentation.java

[9] Representation : Profile

core/src/main/java/org/keycloak/representations/idm/ClientProfileRepresentation.java

[10] Representation : Profiles

core/src/main/java/org/keycloak/representations/idm/ClientProfilesRepresentation.java

[11] Admin REST API : CLI Interface - Policy

Integration/admin-client/src/main/java/org/keycloak/admin/client/resource/ClientPoliciesPoliciesResource.java

[12] Admin REST API : CLI Interface - Profile

Integration/admin-client/src/main/java/org/keycloak/admin/client/resource/ClientPoliciesProfilesResource.java

[13] Context : Interface

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

[14] Event

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

Client Policies Classes

[15] Exception

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

[16] Manager - Interface

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

[17] Client Polices Realm Attributes Keys

server-spi-private/src/main/java/org/keycloak/models/Constants.java

[18] Utility : Convert Policies/Profiles Model to Representation

server-spi-private/src/main/java/org/keycloak/models/utils/ModelToRepresentation.java

[19] Utility : Convert Policies/Profiles Representation to Model

server-spi-private/src/main/java/org/keycloak/models/utils/ModelToRepresentation.java

[20] Manager : Factory - Interface

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

[21] Manager : SPI

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

Client Policies Classes

[22] Vote

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

[23] Condition : Provider - Abstract

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

[24] Condition : Provider - Interface

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

[25] Condition : Provider Factory - Interface

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

[26] Condition : Provider SPI

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

[27] Executor : Provider - Interface

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

[28] Executor : Provider Factory - Interface

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

Client Policies Classes

[29] Executor : Provider SPI

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

[30] Manager/Condition/Executor : Provider SPI Registration

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

[31] Utility : exporting policies and profiles

services/src/main/java/org/keycloak/exportimport/util/ExportUtils.java

[32] Endpoint – Authorization Endpoint

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

[33] Endpoint – Logout Endpoint

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

[34] Endpoint – Token Endpoint

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

[35] Endpoint – Token Introspection Endpoint

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

Client Policies Classes

[36] Endpoint – Token Revocation Endpoint

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

[37] Endpoint – UserInfo Endpoint

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

[38] Keycloak Session – Provides Client Policy Manager.

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

[39] Utility : converting several representations of policies and profiles

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

[40] Model : policy

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

[41] Model : profile

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

[42] Manager : Provider - Default Implementation

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

Client Policies Classes

[43] Manager : Provider Factory - Default Implementation

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

[44] Condition : Provider Implementation – An

services/src/main/java/org/keycloak/services/clientpolicy/condition/AnyClientCondition.java

[45] Condition : Provider Factory Implementation – Any

services/src/main/java/org/keycloak/services/clientpolicy/condition/AnyClientConditionFactory.java

[46] Condition : Provider Implementation – Access Type

services/src/main/java/org/keycloak/services/clientpolicy/condition/ClientAccessTypeCondition.java

[47] Condition : Provider Factory Implementation – Access Type

services/src/main/java/org/keycloak/services/clientpolicy/condition/ClientAccessTypeConditionFactory.java

[48] Condition : Provider Implementation – Role

services/src/main/java/org/keycloak/services/clientpolicy/condition/ClientRolesCondition.java

[49] Condition : Provider Factory Implementation – Role

services/src/main/java/org/keycloak/services/clientpolicy/condition/ClientRolesConditionFactory.java

Client Policies Classes

[50] Condition : Provider Implementation – Scope

services/src/main/java/org/keycloak/services/clientpolicy/condition/ClientScopesCondition.java

[51] Condition : Provider Factory Implementation – Scope

services/src/main/java/org/keycloak/services/clientpolicy/condition/ClientScopesConditionFactory.java

[52] Condition : Provider Implementation – Update Context

services/src/main/java/org/keycloak/services/clientpolicy/condition/ClientUpdaterContextCondition.java

[53] Condition : Provider Factory Implementation – Update Context

services/src/main/java/org/keycloak/services/clientpolicy/condition/ClientUpdaterContextConditionFactory.java

[54] Condition : Provider Implementation – Update Source Group

services/src/main/java/org/keycloak/services/clientpolicy/condition/ClientUpdaterSourceGroupsCondition.java

[55] Condition : Provider Factory Implementation – Update Source Group

services/src/main/java/org/keycloak/services/clientpolicy/condition/ClientUpdaterSourceGroupsCondition.java

[56] Condition : Provider Implementation – Update Source Host

services/src/main/java/org/keycloak/services/clientpolicy/condition/ClientUpdaterSourceHostsCondition.java

Client Policies Classes

[57] Condition : Provider Factory Implementation – Update Source Host

services/src/main/java/org/keycloak/services/clientpolicy/condition/ClientUpdaterSourceHostsConditionFactory.java

[58] Condition : Provider Implementation – Update Source Role

services/src/main/java/org/keycloak/services/clientpolicy/condition/ClientUpdaterSourceRolesCondition.java

[59] Condition : Provider Factory Implementation – Update Source Role

services/src/main/java/org/keycloak/services/clientpolicy/condition/ClientUpdaterSourceRolesConditionFactory.java

[60] Context : CRUD by Admin REST API - Abstract

services/src/main/java/org/keycloak/services/clientpolicy/context/AbstractAdminClientCRUDContext.java

[61] Context : CRUD by Dynamic Client Registration - Abstract

services/src/main/java/org/keycloak/services/clientpolicy/context/AbstractDynamicClientCRUDContext.java

[62] Context : CRUD by Admin REST API - Register

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

[63] Context : CRUD by Admin REST API - Registered

services/src/main/java/org/keycloak/services/clientpolicy/context/AdminClientRegisteredContext.java

Client Policies Classes

[64] Context : CRUD by Admin REST API - Unregister

services/src/main/java/org/keycloak/services/clientpolicy/context/AdminClientUnregisterContext.java

[65] Context : CRUD by Admin REST API - Update

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

[66] Context : CRUD by Admin REST API - Updated

services/src/main/java/org/keycloak/services/clientpolicy/context/AdminClientUpdatedContext.java

[67] Context : CRUD by Admin REST API - View

services/src/main/java/org/keycloak/services/clientpolicy/context/AdminClientViewContext.java

[68] Context : Authorization Request

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

[69] Context : CRUD - Interface

services/src/main/java/org/keycloak/services/clientpolicy/context/ClientCRUDContext.java

[70] Context : CRUD by Dynamic Client Registration - Register

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

Client Policies Classes

[71] Context : CRUD by Dynamic Client Registration - Registered

services/src/main/java/org/keycloak/services/clientpolicy/context/DynamicClientRegisteredContext.java

[72] Context : CRUD by Dynamic Client Registration - Unregister

services/src/main/java/org/keycloak/services/clientpolicy/context/DynamicClientUnregisterContext.java

[73] Context : CRUD by Dynamic Client Registration - Update

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

[74] Context : CRUD by Dynamic Client Registration - Updated

services/src/main/java/org/keycloak/services/clientpolicy/context/DynamicClientUpdatedContext.java

[75] Context : CRUD by Dynamic Client Registration - View

services/src/main/java/org/keycloak/services/clientpolicy/context/DynamicClientViewContext.java

[76] Context : Logout

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

[77] Context : Token Introspect

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

Client Policies Classes

[78] Context : Token Refresh

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

[79] Context : Token Request

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

[80] Context : Token Revoke

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

[81] Context : User Info Request

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

[82] Executor : Provider Implementation – Accept Confidential

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

[83] Executor : Provider Factory Implementation – Accept Confidential

services/src/main/java/org/keycloak/services/clientpolicy/condition/ConfidentialClientAcceptExecutorFactory.java

[84] Executor : Provider Implementation – Require Consent

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

Client Policies Classes

[85] Executor : Provider Implementation Factory – Require Consent

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

[86] Executor : Provider Implementation – Enforce Holder-of-Key bound token

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

[87] Executor : Provider Implementation Factory – Enforce Holder-of-Key bound token

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

[88] Executor : Provider Implementation – Enforce Proof key for code exchange (PKCE)

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

[89] Executor : Provider Implementation Factory – Enforce Proof key for code exchange (PKCE)

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

[90] Executor : Provider Implementation – Accept/Enforce secure client authentication method

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

[91] Executor : Provider Implementation Factory – Accept/Enforce secure client authentication method

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

Client Policies Classes

[92] Executor : Provider Implementation – Accept secure URI

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

[93] Executor : Provider Implementation Factory – Accept secure URI

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

[94] Executor : Provider Implementation – Accept secure request object

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

[95] Executor : Provider Implementation Factory – Accept secure request object

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

[96] Executor : Provider Implementation – Accept secure response type

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

[97] Executor : Provider Implementation Factory – Accept secure response type

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

[98] Executor : Provider Implementation – Accept secure session

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

Client Policies Classes

[99] Executor : Provider Implementation Factory – Accept secure session

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

[100] Executor : Provider Implementation – Accept/Enforce secure signature algorithm to client's signed data

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

[101] Executor : Provider Implementation Factory – Accept/Enforce secure signature algorithm to client's signed data

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

[102] Executor : Provider Implementation – Accept/Enforce secure signature algorithm to JWT authentication

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

[103] Executor : Provider Implementation Factory – Accept/Enforce secure signature algorithm to JWT authentication

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

[104] Endpoint – Dynamic Client Registration/Update

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

[105] Realm Manager : setup client policies

services/src/main/java/org/keycloak/services/managers/RealmManager.java

Client Policies Classes

[106] Endpoint – Admin REST API for Client Update

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

[107] Endpoint – Admin REST API for Client Registration

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

[108] Admin REST API - policies

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

[109] Admin REST API - profiles

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

[110] Realm Admin Resource – import/export policies/profiles

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

[111] Realm Admin Resource – import/export policies/profiles

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

[112] Policy Implementation Registration

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

Client Policies Classes

[113] Global Profiles

services/src/main/resources/keycloak-default-client-profiles.json

[114] Testsuite : Run Helper – set proper realm

testsuite/integration-arquillian/servers/auth-server/services/testsuite-providers/src/main/java/org/keycloak/testsuite/runonserver/RunHelpers.java

[115] Testsuite : 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

[116] Testsuite : Condition – 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/condition/TestRaiseExeptionConditionFactory.java

[117] Testsuite : Executor – 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/executor/TestRaiseExeptionExecutor.java

[118] Testsuite : Executor – 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/executor/TestRaiseExeptionExecutorFactory.java

Client Policies Classes

[119] Testsuite : 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

[120] Testsuite : 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

[121] Testsuite : Arquillian Integration Test –Abstract

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

[122] Testsuite : Arquillian Integration Test – Import/Export policies/profiles

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

[123] Testsuite : Arquillian Integration Test – Load/Update policies/profiles

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

[124] Arquillian Integration Test – Client Policy Basics

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

Client Policies Classes

[125] Testsuite : Internal Component Representation – set proper realm

testsuite/integration-arquillian/tests/base/src/test/java/org/keycloak/testsuite/runonserver/InternalComponentRepresentation.java

[126] Testsuite : Run on Server Test – set proper realm

testsuite/integration-arquillian/tests/base/src/test/java/org/keycloak/testsuite/runonserver/RunOnServerTest.java

[127] Testsuite : Arquillian Integration Test –Log Level Control

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

[128] Testsuite : Arquillian Integration Test –Log Level Control

testsuite/utils/src/main/resources/log4j.properties

[129] Admin Console UI : message definition

themes/src/main/resources/theme/base/admin/messages/admin-messages_en.properties

[130] Admin Console UI : Application – add route

themes/src/main/resources/theme/base/admin/resources/js/app.js

[131] Admin Console UI : Realm Controller – define functions

themes/src/main/resources/theme/base/admin/resources/js/controllers/realm.js

Client Policies Classes

[132] Admin Console UI : Loader – define loader

themes/src/main/resources/theme/base/admin/resources/partials/client-policies-list.html

[133] Admin Console UI : Service – define external service

themes/src/main/resources/theme/base/admin/resources/partials/client-policies-policy-edit-condition.html

[134] Admin Console UI : Client Polices – page

themes/src/main/resources/theme/base/admin/resources/partials/client-policies-policy-edit.html

[135] Admin Console UI : Client Polices – page

themes/src/main/resources/theme/base/admin/resources/partials/client-policies-profiles-edit-executor.html

[136] Admin Console UI : Client Polices – page

themes/src/main/resources/theme/base/admin/resources/partials/client-policies-profiles-edit.html

[137] Admin Console UI : Client Polices – page

themes/src/main/resources/theme/base/admin/resources/partials/client-policies-profiles-json.html

[138] Admin Console UI : Client Polices – page

themes/src/main/resources/theme/base/admin/resources/partials/client-policies-profiles-list.html

Client Policies Classes

[139] Admin Console UI : Client Polices – templete

themes/src/main/resources/theme/base/admin/resources/templates/kc-menu.html

[140] Admin Console UI : Client Polices – templete

themes/src/main/resources/theme/base/admin/resources/templates/kc-provider-config.html

[141] Admin Console UI : Client Polices – templete

themes/src/main/resources/theme/base/admin/resources/templates/kc-tabs-realm.html

End