CIBA Implementation Practical Guide

For keycloak FAPI-SIG

Jun 2021

PROPOSED DRAFT

Preface

This document describes internals of Client Initiated Backchannel Authentication (CIBA*1) support. This support was introduced from keycloak 12 and has been developed based on its design document*2 .

The aim of this document is as follows.

- making it ease for programmers to comprehend codes for CIBA support.

Target readers of this document are as follows.

- Reviewers who reviews codes for CIBA support.
- Programmers who want to make some CIBA support related contribution.

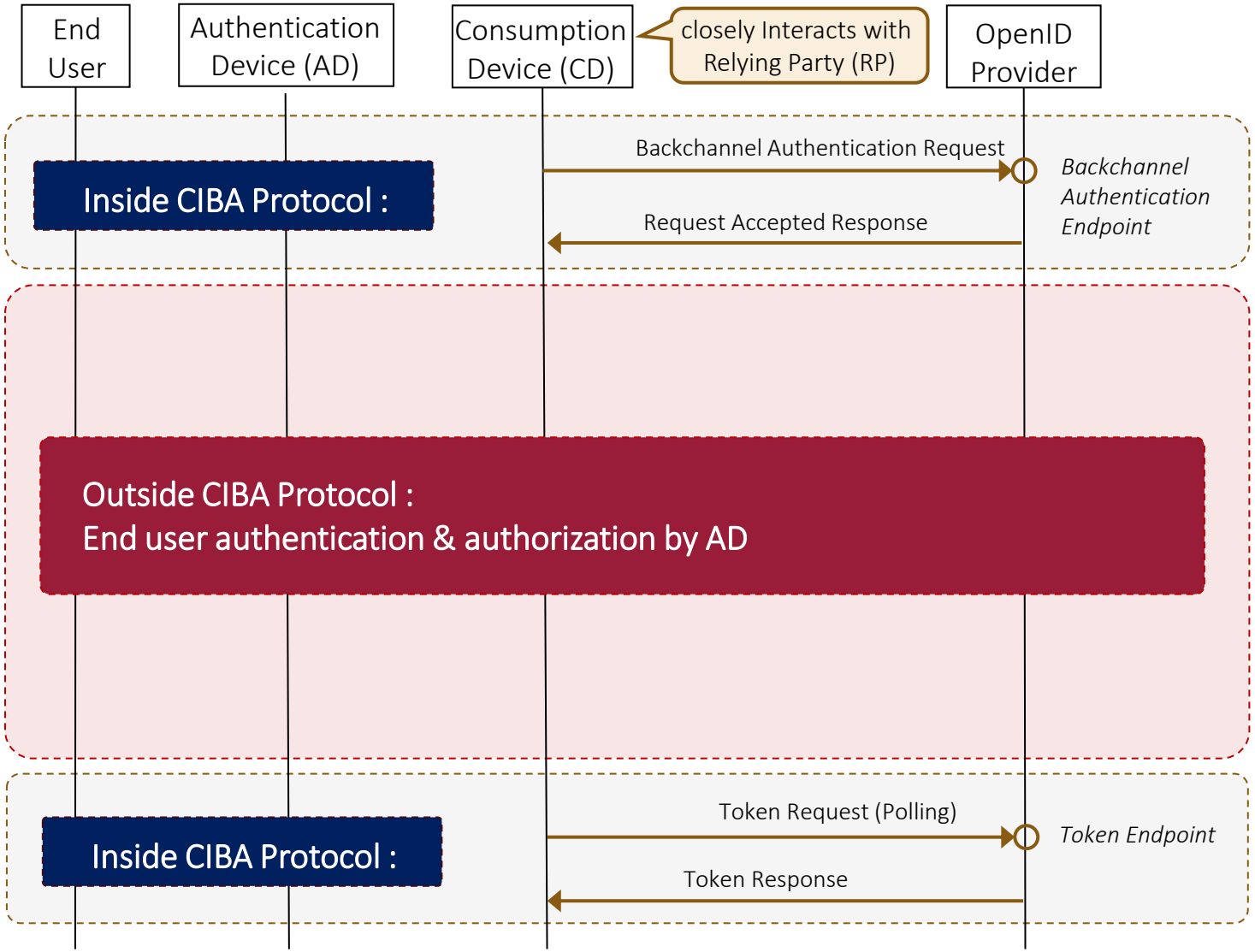
*1 : https://openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html

*2 : https://github.com/keycloak/keycloak-community/blob/master/design/client-initiated-backchannel-authentication-flow.md

Table of Contents

Overview

CIBA Flow : Protocol specified part only



CIBA specification does not specify how to do end user Authentication(AuthN) & Authorization(AuthZ) by Authentication Device(AD).

To implement CIBA flow, we also need to specify this part (Outside CIBA Protocol) and implement it.

This part is case by case so that I could not implement it onto keycloak's codebase itself. Therefore, I've decided that the way of doing AuthN & AuthZ by AD can be implemented as providers and I've defined the interface of AuthN & AuthZ by AD these providers need to follow. This interface does not depend on a specific way of doing it.

PROPOSED DRAFT

CIBA Flow : Outside CIBA part - High Level Design

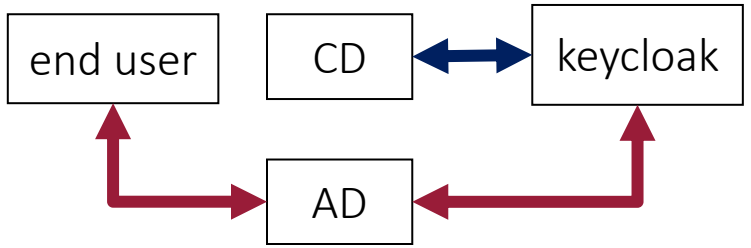
To accommodate broader range of ADs, outside CIBA part should not depend on the specific way of AuthN and AuthZ by AD.

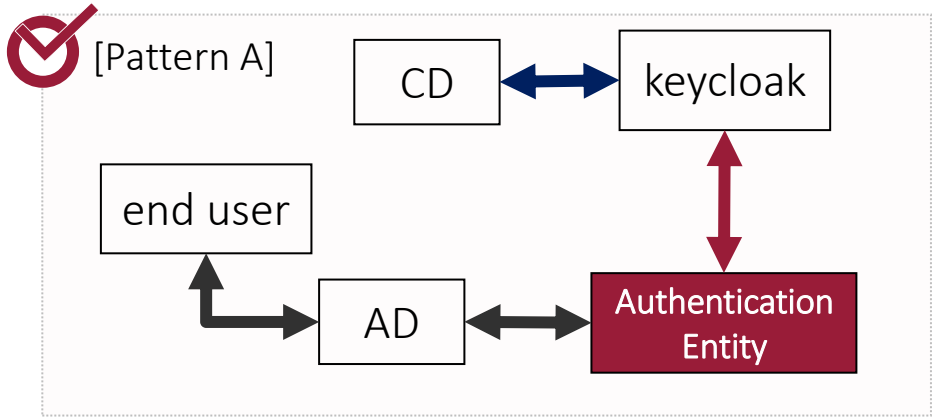
To do so, there are two patterns, A and B, depending on where interfaces between inside CIBA part and outside CIBA part exists.

Pattern A add a new entity “Authentication Entity” outside keycloak while pattern B add it inside keycloak.

Pattern B’s interface should take the form of some method’s signature while pattern A’ s interface can take the form of common REST API style.

Considering extensibility, pattern A has been taken.

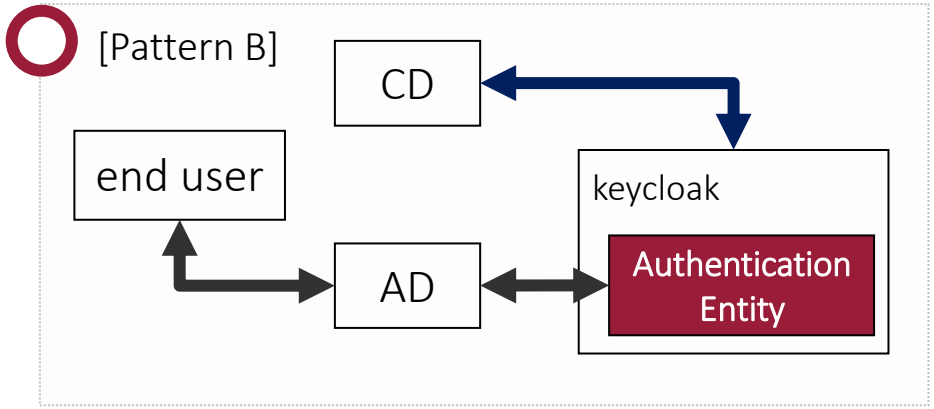




: Inside CIBA Protocol

: Outside CIBA Protocol

: Outside CIBA Protocol
AD specific part



CIBA Flow : Interfaces on outside CIBA part

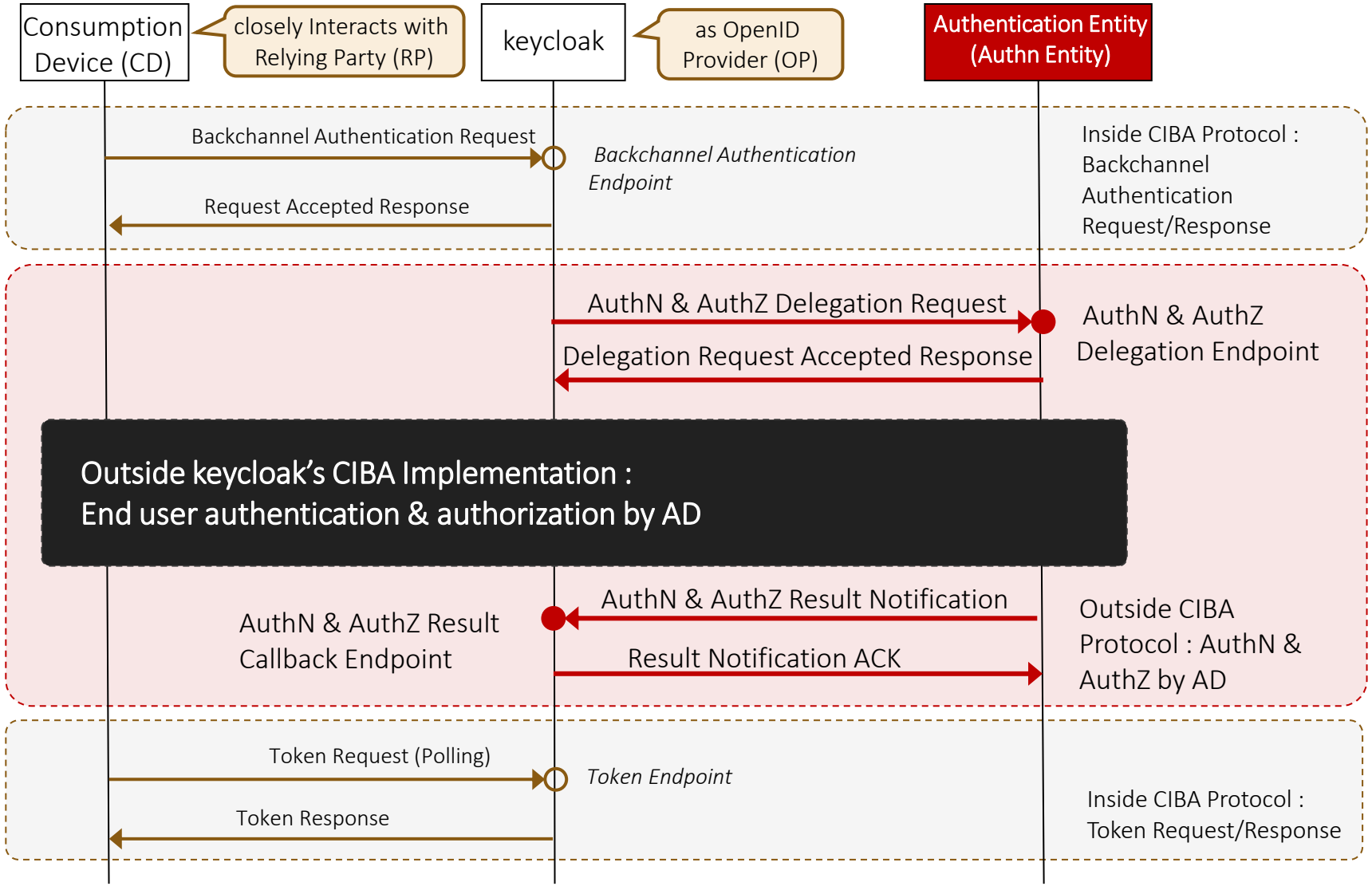
Keycloak itself does not do AuthN & AuthZ byAD. It is delegated to the other entity called “Authentication Entity (Authn Entity)”.

This CIBA support does not treat any kind of the specific way of AuthN & AuthZ by AD. It is up to actual implementation of an authn entity.

This CIBA support only defines the interfaces between keycloak and an authn entity.

However, to confirm this prototype works, the reference implementation of the authn entity was provided :

https://github.com/tnorimat/ciba-authentication-entity

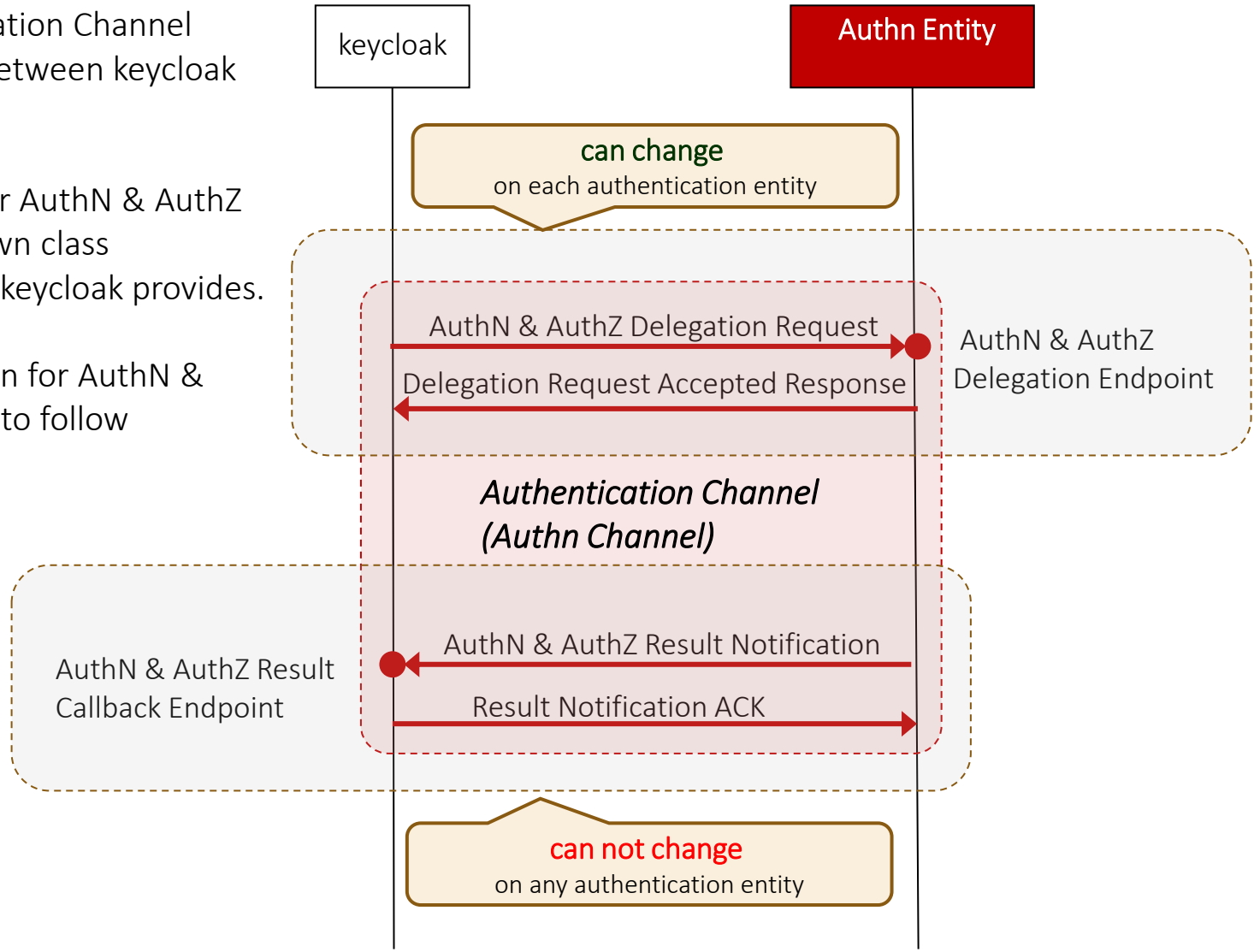


CIBA Flow : Interfaces on outside CIBA part

Keycloak provides an interface called “Authentication Channel (Authn Channel)” abstracting communications between keycloak and an authn entity.

Programmers can change the implementation for AuthN & AuthZ Delegation Request/Response by writing their own class implementing “AuthenticationChannelProvider” keycloak provides.

Programmers can not change the implementation for AuthN & AuthZ Result Notification/ACK so that they need to follow keycloak’s implementation.



CIBA Flow : Client Authentication

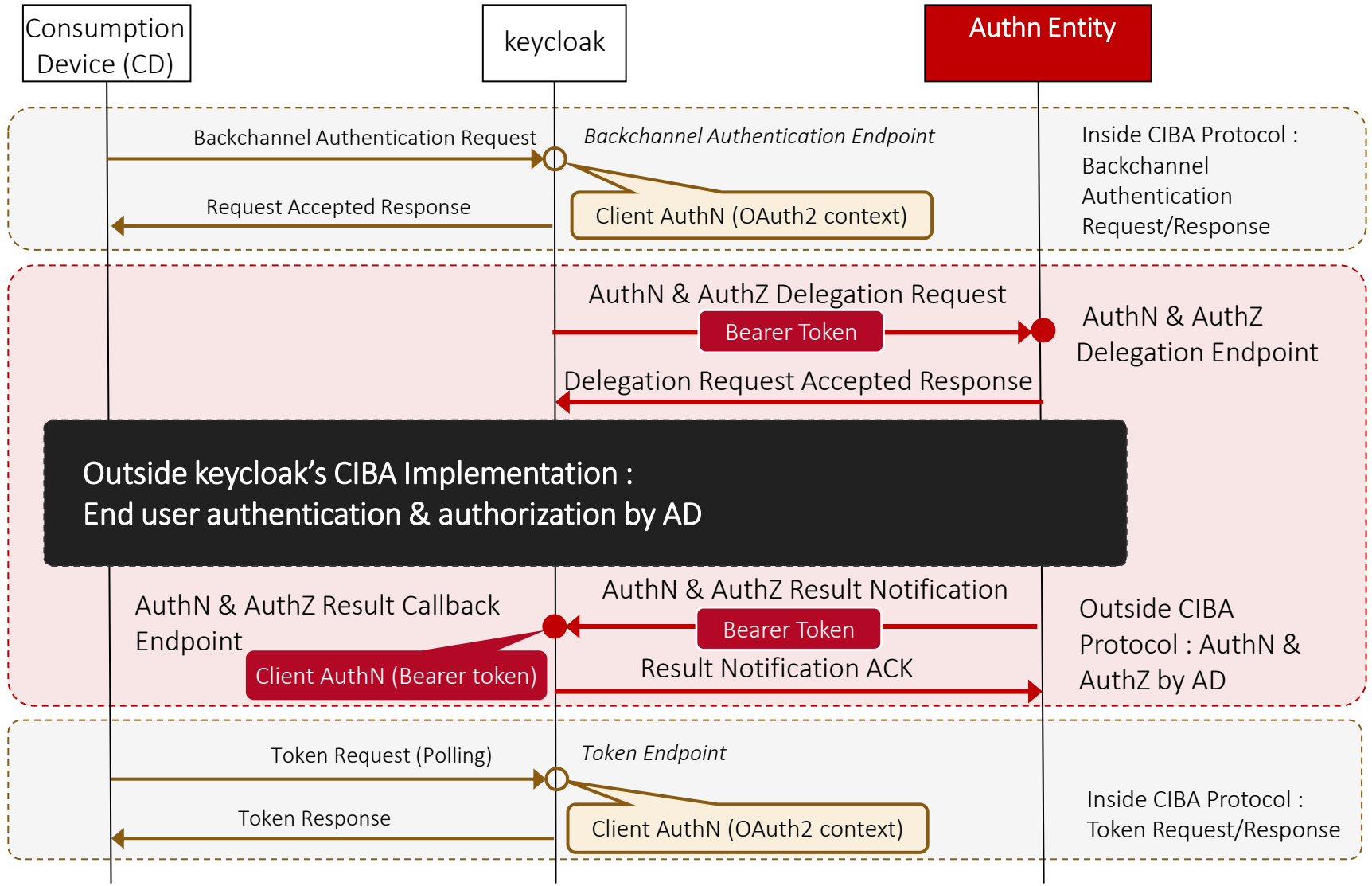
[Inside CIBA : Authenticate a CD]

Keycloak authenticates a client (CD) as OAuth2's client on the following endpoint :

- Backchannel Authentication Endpoint
- Token Endpoint

[Outside CIBA : Authenticate an Authn Entity]

Keycloak dose not treat an authn entity as OAuth2's client. Instead of that, keycloak employs bearer token authentication following RFC 6750 using Authorization request header field.



PROPOSED DRAFT

Entities

Entities in CIBA flow are the followings :

[Inside CIBA protocol]

- End User
- Authentication Device (AD)
- Consumption Device (CD)
- keycloak (OpenID Provider)

[Outside CIBA protocol]

- Authentication Entity (Authn Entity)

Scope

Scope

This document covers the followings :

- Specification on Inside CIBA Protocol
- Specification on Outside CIBA Protocol
- Endpoint specification of CIBA Flow
- HTTP Request/Response specification of CIBA Flow
- keycloak’s version : 12*

Especially, this document does not cover the followings :

- Performance
- Usability

* : CIBA’s implementation might be changed after keycloak 12 where first CIBA support was introduced.

Functional Specification

Functional Specification

[Inside CIBA Protocol]

- Backchannel Authentication Request
  - Parameter used to identify an end-user : *login_hint*
  - Value of *login_hint* : *username* in keycloak
  - Supported parameters : *scope*, *binding_message*
- Token Request
  - Mode : poll
  - Expiration of *auth_req_id* : supported (by *expires_in*)
  - Request throttling : supported (by *interval*)

Functional Specification

[Outside CIBA Protocol]

- AuthN & AuthZ by AD
  - The way of doing it : Delegating it to external Authn Entity
  - The type of its communication : asynchronous
  - ID of an end-user : *username* in keycloak
  - Authorization : supported (whether it is required or not is notified from keycloak)
- Supported features by issued tokens
  - Token Refresh
  - Token Introspection
  - Token Revocation
  - User Info Request
  - Logout

Prerequisite

Prerequisite

[User]

Users AuthN & AuthZ by AD must be registered on keycloak in advance.

[CD]

CD must be registered on keycloak as a confidential client in advance.

You need to prepare it by yourself. (e.g. curl, postman)

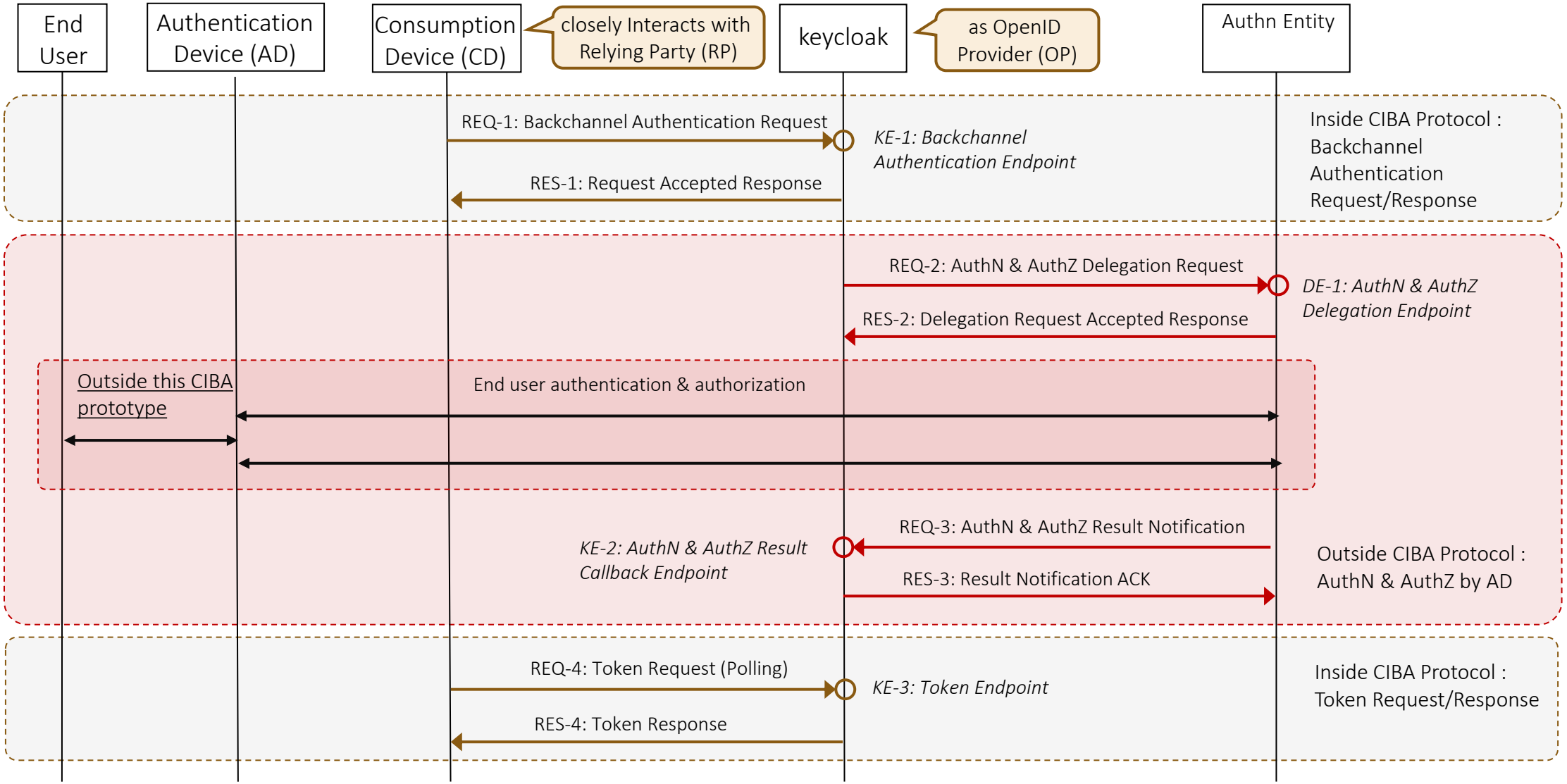
[Authn Entity]

You need to prepare an authn entity by yourself.
You could use its reference implementation*1.

*1 : https://github.com/tnorimat/ciba-authn-entity

Interface Specification

CIBA Flow : Endpoints and Messages



PROPOSED DRAFT

CIBA Flow Overview

CIBA flow consists the following 2 parts:

- Inside CIBA Protocol (defined by CIBA specification)
  - Backchannel Authentication Request/Response
  - Token Request/Response
- Outside CIBA Protocol (NOT defined by CIBA specification)
  - AuthN & AuthZ by AD Request/Response

The order of running these part in CIBA flowis as follows :

1. Inside CIBA Protocol : Backchannel Authentication Request/Response
2. Outside CIBA Protocol : AuthN & AuthZ by AD Request/Response
3. Inside CIBA Protocol : Token Request/Response

Endpoints

Endpoints in CIBA protocol are the followings :

- On keycloak
  - KE-1: Backchannel Authentication Endpoint

    CD sends a backchannel authentication request to it.

    This endpoint is defined by CIBA specification.
  - KE-2: AuthN & AuthZ Result Callback Endpoint

    Authn entity sends the result of AuthN & AuthZ by AD to it.

    This endpoint is not defined by CIBA specification.
  - KE-3: Token Endpoint

    CD sends a token request to it.

    This endpoint is defined by OAuth2 and CIBA specification.
- On Authn Entity
  - DE-1: AuthN & AuthZ Delegation Endpoint

    Keycloak sends AuthN & AuthZ by AD delegation request to it.

    This endpoint is not defined by CIBA specification.

KE-1: Backchannel Authentication Endpoint

[Overview]

Keycloak plays a role as HTTP Server while CD as HTTP Client.

CD sends a backchannel authentication request to keycloak.

Keycloak returns *auth_req_id* that identifies the corresponding CIBA flow.

CD uses it for token request afterwards.

If keycloak returns an abnormal response, the corresponding CIBA flow is aborted.

<URI>

http(s)://{host}:{port}/auth/realms/{realm}/protocol/openid-connect/ext/ciba/auth

<Authentication>

Required (Basic Authentication with *client_id* and *client_secret* as default)

REQ-1: Backchannel Authentication Request

<Method> : POST

<Content-Type> : application/x-www-form-urlencoded

<Parameters>

login_hint : REQUIRED

It identifies the end user for AuthN and AuthZ by AD.

Its value must be “username” or “email” of the user registered in keycloak.

scope : REQUIRED

“scope” parameter defined by OAuth2 specification.

binding_message : OPTIONAL

Its value is intended to be shown in both CD and AD’s UI.

RES-1: Request Accepted Response

[Normal Case]

<Status Code>

200 OK

<Content-Type>

application/json

<Parameters>

auth_req_id : REQUIRED

It identifies the CIBA flow. It can be used for token request.

expires_in : REQUIRED

It expresses the expiration time in sec for auth_req_id.

interval : OPTIONAL

It shows the interval for which CD needs to wait for token request.

RES-1: Request Accepted Response

[Abnormal Case 1]

<Case> : CD's client authentication failed.

<Status Code> : 401 Unauthorized

<Content-Type> : application/json

<Entities>

error : "unauthorized_client"

error_description : "invalid client secret"

[Abnormal Case 2]

<Case> : CD is not registered as a confidential client.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entities>

error : "unauthorized_client"

error_description : "INVALID_CREDENTIALS: Invalid client credentials"

RES-1: Request Accepted Response

[Abnormal Case 3]

<Case> : CD is registered as a confidential client but deactivated.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entities>

error : “unauthorized_client”

error_description : “Invalid client credentials”

[Abnormal Case 4]

<Case> : Required parameter “scope” is missing.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entities>

error : “invalid_request”

error_description : “missing parameter : scope”

RES-1: Request Accepted Response

[Abnormal Case 5]

<Case> : Required parameter “login_hint” is missing.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entities>

error : “invalid_request”

error_description : “missing parameter : login_hint”

[Abnormal Case 6]

<Case> : The user specified by “login_hint” does not exist.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entities>

error : “unknown_user_id”

error_description : “no user found”

RES-1: Request Accepted Response

[Abnormal Case 7]

<Case> : The user specified by “login_hint” is deactivated.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entities>

error : “unknown_user_id”

error_description : “user deactivated”

[Abnormal Case 8]

<Case> : not supported parameter “client_notification_token” is specified.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entities>

error : “invalid_request”

error_description : “Ping and push modes not supported. Use poll mode instead.”

RES-1: Request Accepted Response

[Abnormal Case 9]

<Case> : not supported parameter “user_code” is specified.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entities>

error : “invalid_request”

error_description : “User code not supported""

[Abnormal Case 10]

<Case> : Something unexpected error happened.

<Status Code> : 500 Internal Server Error

DE-1: AuthN & AuthZ Delegation Endpoint

[Overview]

Authn Entity plays a role as HTTP Server while keycloak as HTTP Client.

keycloak sends an AuthN & AuthZ delegation request to Authn Entity.

Authn Entity returns the result of AuthN & AuthZ by AD with JWT Bearer Token to identify afterwards.

This endpoint is not defined by CIBA specification.

If an authentication entity returns an abnormal response, the corresponding CIBA is aborted.

<URI>

depending on your using authentication entity.

<Authentication>

Nothing

REQ-2: AuthN & AuthZ Delegation Request

<Method> : POST

<Content-Type> : application/x-www-form-urlencoded

<Authorization> : Bearer [JWT Bearer Token]

It identifies the flow of AuthN & AuthZ by AD in Authn Entity.

<Parameters>

loginHint : REQUIRED

It identifies the end user for AuthN and AuthZ by AD.

Its value must be “username” of the user registered in keycloak.

scope : REQUIRED

“scope” parameter defined by OAuth2 specification.

consentRequired : REQUIRED

It shows whether an authn entity needs to get consent from the end user about scope.

acrValue : OPTIONAL

It indicates ACRs CD wants to apply to AuthN and AuthZ by AD.

bindingMessage : OPTIONAL

Its value is intended to be shown in both CD and AD’s UI.

RES-2: Delegation Request Accepted Response

[Normal Case]

<Status Code> : 201 Created

[Abnormal Case 1]

<Case> : Invalid input

<Status Code> : 400 Bad Request

[Abnormal Case 2]

<Case> : Something unexpected error happened in an authn entity.

<Status Code> : 500 Internal Server Error

KE-2: AuthN & AuthZ Result Callback Endpoint

[Overview]

keycloak plays a role as HTTP Server while an authn entity as HTTP Client.
The authn entity sends the result of AuthN & AuthZ by AD to keycloak with JWT bearer token to identify the context of AuthN & AuthZ by AD.

This endpoint is not defined by CIBA specification.

<URI>

http(s)://{host}:{port}/auth/realms/{realm}/protocol
/openid-connect/ext/ciba/auth/callback

<Authentication>

Using HTTP Authorization header.

Authorization: Bearer <JWT bearer token>

JWT bearer token : token sent from keycloak in REQ-2 AuthN & AuthZ Delegation Request

REQ-3: AuthN & AuthZ Result Notification

<Method> : POST

<Content-Type> : application/x-www-form-urlencoded

<Authorization> : Bearer [JWT Bearer Token]

It identifies the flow of AuthN & AuthZ by AD in Authentication Entity.

<Parameters>

status : REQUIRED

The result of AuthN and AuthZ by AD.

SUCCEEDED : Both AuthN and AuthZ have succeeded. (only AuthN if AuthZ is not required)

UNAUTHORIZED : AuthN has succeeded but AuthZ has been denied.

CANCELLED : AuthN have been cancelled.

RES-3: Result Notification ACK

[Normal Case]

<Status Code> : 200 OK

KE-3: Token Endpoint

[Overview]

Keycloak plays a role as HTTP Server while CD as HTTP Client.

CD sends a token request to keycloak with auth_req_id that identifies the corresponding CIBA flow.

If keycloak returns an abnormal response, the corresponding CIBA is aborted.

<URI>

http(s)://{host}:{port}/auth/realms/{realm}/protocol/openid-connect/token

<Authentication>

Required (Basic Authentication with client_id and client_secret as default)

REQ-4: Token Request (Polling)

<Method> : POST

<Content-Type> : application/x-www-form-urlencoded

<Parameters>

grant_type : REQUIRED

It must be “urn:openid:params:grant-type:ciba”.

auth_req_id : REQUIRED

It identifies the CIBA flow.

RES-4: Token Response

[Normal Case]

<Status Code> : 200 OK

<Content-Type> : application/json

<Parameters>

access_token: REQUIRED

Access token defined by OAuth2 specification.

expires_in : REQUIRED

Access token’s expiration time defined by OAuth2 specification.

token_type : REQUIRED

Token type defined by OAuth2 specification. Its value is “bearer”.

scope : OPTIONAL

Scope defined by OAuth2 specification.

refresh_token : OPTIONAL

Refresh token defined by OAuth2 specification.

refresh_expires_in : OPTIONAL

Refresh token’s expiration time.

id_token : OPTIONAL

ID token defined by OIDC specification.

RES-4: Token Response

[Abnormal Case 1]

<Case> : CD's client authentication failed.

<Status Code> : 401 Unauthorized

<Content-Type> : application/json

<Entity>

error : "unauthorized_client"

error_description : "invalid client secret"

[Abnormal Case 2]

<Case> : auth_req_id is missing.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entity>

error : "invalid_request"

error_description : "Missing parameter: auth_req_id"

RES-4: Token Response

[Abnormal Case 3]

<Case> : auth_req_id format is invalid.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entity>

error : “invalid_grant”

error_description : “Invalid Auth Req ID”

[Abnormal Case 4]

<Case> : auth_req_id has already been used.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entity>

error : “invalid_grant”

error_description : “Invalid Auth Req ID”

RES-4: Token Response

[Abnormal Case 5]

<Case> : auth_req_id has not yet been issued.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entity>

error : “invalid_grant”

error_description : “Invalid Auth Req ID”

[Abnormal Case 6]

<Case> : auth_req_id has already expired.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entity>

error : “expired_token”

error_description : “Auth Req ID has expired.”

RES-4: Token Response

[Abnormal Case 7]

<Case> : CD send a request without waiting for the time specified by the parameter “interval”.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entity>

error : “slow_down”

error_description : “Too early to access.”

Notes : add +5 sec to the interval as the penalty for too much early access.

[Abnormal Case 8]

<Case> : AuthN & AuthZ by AD has not yet been completed.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entity>

error : “authorization_pending”

error_description : “The authorization request is still pending as the end-user hasn’t yet been authenticated.”

RES-4: Token Response

[Abnormal Case 9]

<Case> : AuthN & AuthZ by AD has been time out.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entity>

error : “access_denied”

error_description : “authentication timed out.”

[Abnormal Case 10]

<Case> : AuthN & AuthZ by AD has failed.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entity>

error : “access_denied”

error_description : “authentication failed.”

RES-4: Token Response

[Abnormal Case 11]

<Case> : AuthN & AuthZ by AD has been cancelled.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entity>

error : “access_denied”

error_description : “authentication cancelled.”

[Abnormal Case 12]

<Case> : AuthZ by AD has been denied.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entity>

error : “access_denied”

error_description : “not authorized.”

RES-4: Token Response

[Abnormal Case 13]

<Case> : Unexpected error happened on AuthN & AuthZ by AD.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entity>

error : “invalid_grant”

error_description : “unknown authentication result.”

[Abnormal Case 14]

<Case> : AuthN & AuthZ by AD has been succeeded but the creation of corresponding user session failed.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entity>

error : “invalid_grant”

error_description : “user session not found.”

RES-4: Token Response

[Abnormal Case 15]

<Case> : Different user that CD does not required to be authenticated has been authenticated by AD.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

<Entity>

error : “invalid_grant”

error_description : “different user authenticated.”

[Abnormal Case 16]

<Case> : Different CD that keycloak did not send auth_req_id sends a token request.

<Status Code> : 400 Bad Request

<Content-Type> : application/json

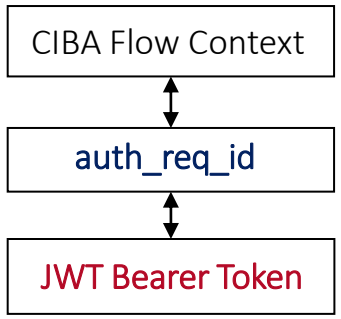
<Entity>

error : “invalid_grant”

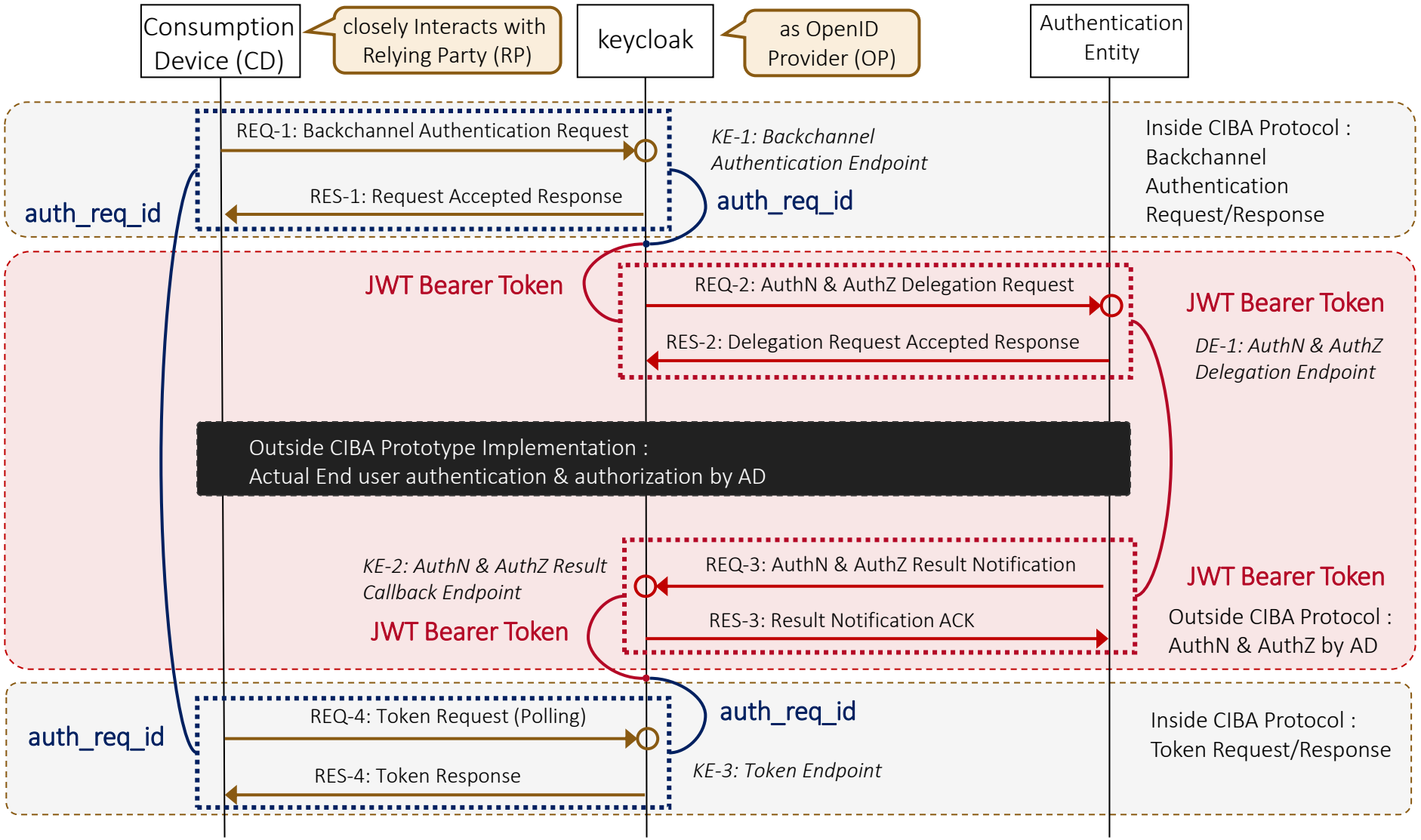
error_description : “unauthorized client.”

Internals

CIBA Flow : Session Binding

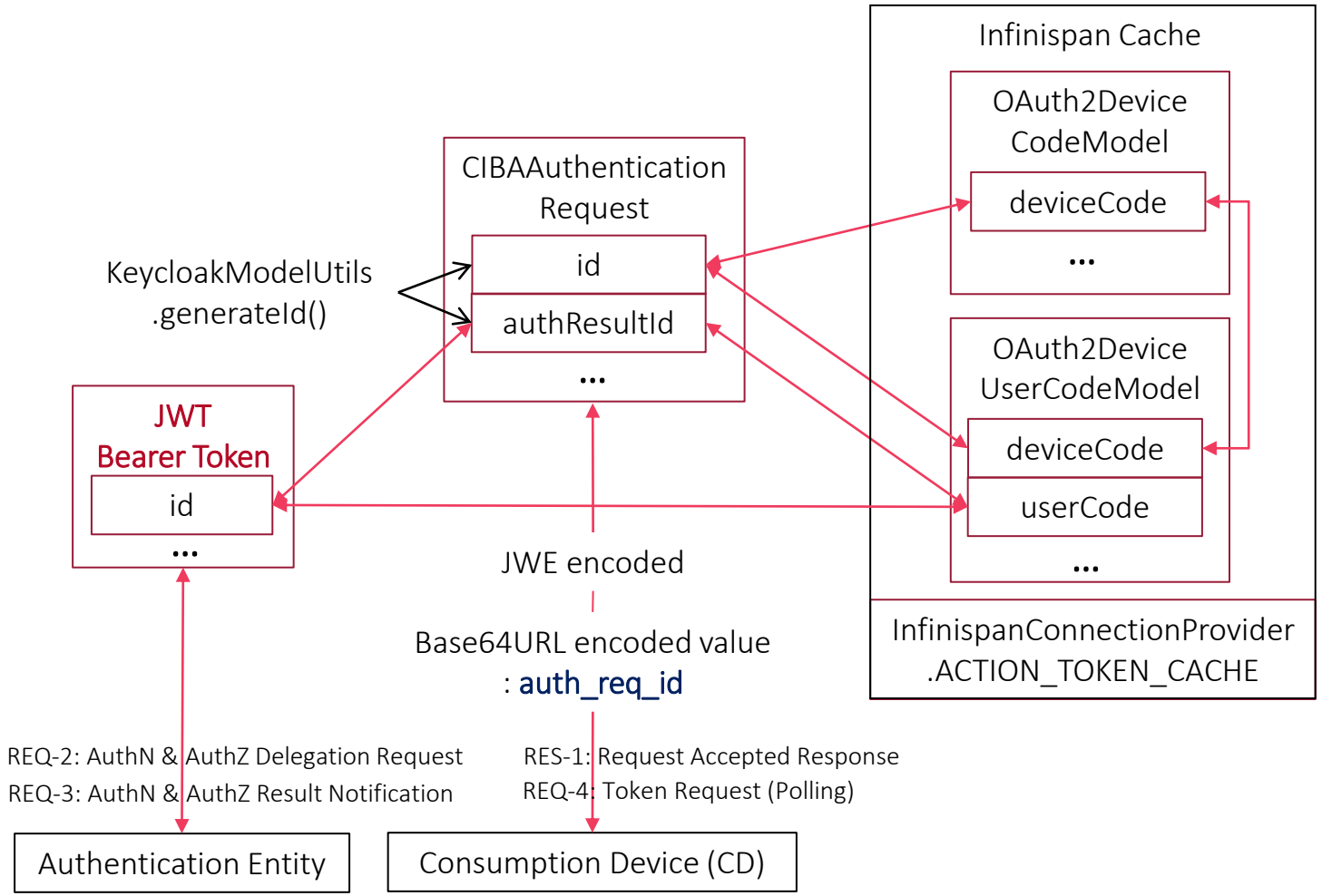


To identify the CIBA flow between keycloak and CD, auth_req_id is used.
To identity the CIBA flow between keycloak and Authentication Entity, JWT Bearer Token is introduced.
Keycloak retains the relationship between auth_req_id and JWT Bearer Token.



PROPOSED DRAFT

CIBA Flow : Session Binding



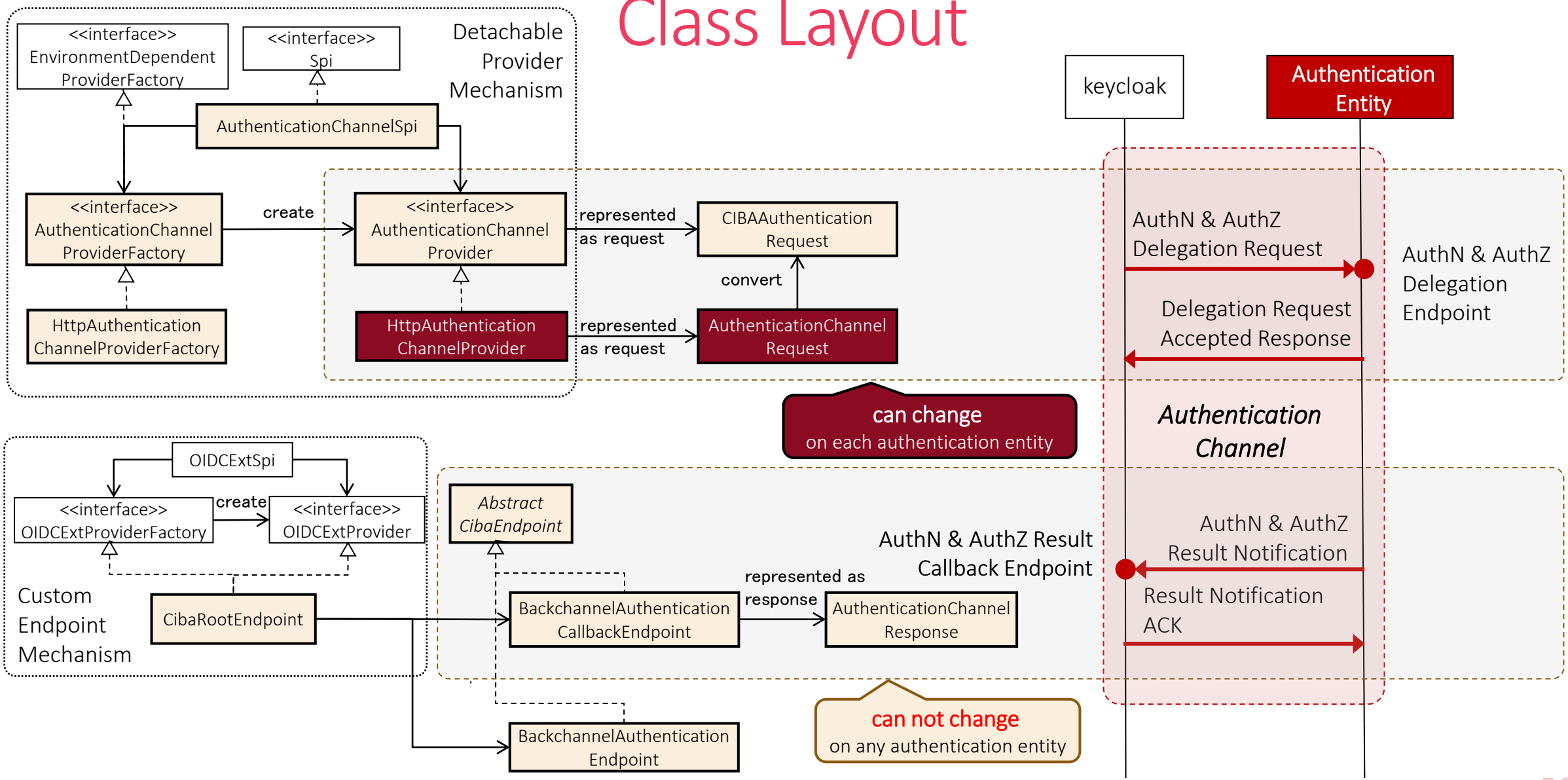
To bind separate sessions between CD/keycloak and Authentication Entity/keycloak, keycloak uses **authResultID.**

JWE encoded auth_req_id is used between keycloak and CD. It includes **authResultID.**

JWT Bearer Token is used between keycloak and Authentication Entity that includes **authResultID.**

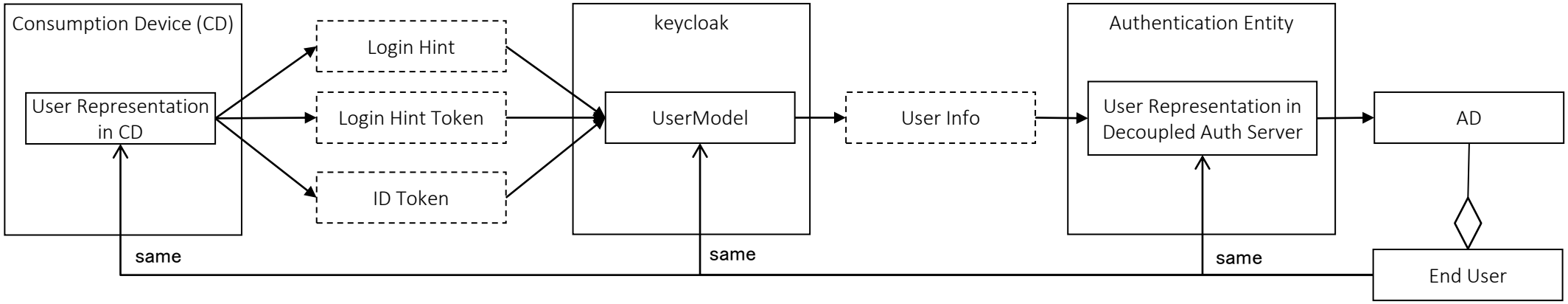
keycloak stores CIBA Flow's session context data whose access key is **authResultID**. Device Grant's cache mechanism is re-used because both Device Grant and CIBA Flow are decoupled flows and share some characteristics.

CIBA Flow : Interfaces on non-protocol specified part Class Layout



PROPOSED DRAFT

User Resolver

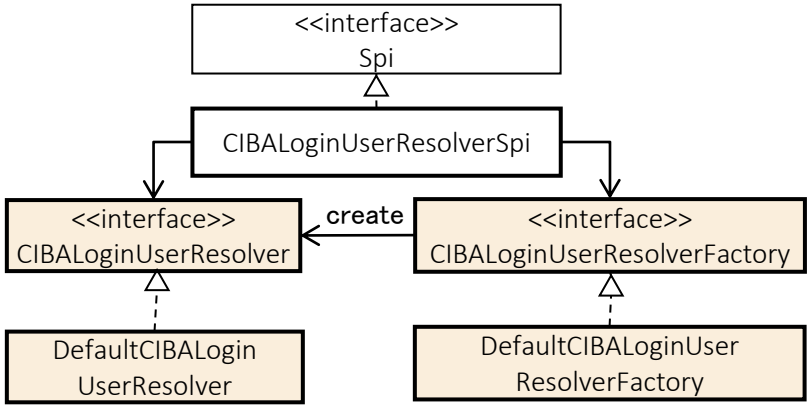


It is possible that each entities included in CIBA flow identifies the same user by the different way.

Also, it is possible that the conveyance of the information that is used for identifying the user the takes the different forms.

Considering these points, this CIBA prototype implementation provides “User Resolver”. It can convert each user representation and format used between CD and keycloak, keycloak and Authentication Entity.

Developer can implement its own User Resolver.



Trial Run

Run by Arquillian Integration Test

To confirm that this CIBA prototype implementation works, run the corresponding Arquillian Integration Test (org.keycloak.testsuite.client.CIBATest)

```
> mvn -f testsuite/integration-arquillian/tests/base/pom.xml test
  -Dtest=org.keycloak.testsuite.client.CIBATest
```

Run with Decoupled Auth Server Reference Implementation

Add config for Authentication Entity on the keycloak config file.

- AuthN & AuthZ Delegation Endpoint

standalone.xml :

```
<spi name="ciba-auth-channel">
   <default-provider>ciba-http-auth-channel</default-provider>
   <provider name="ciba-http-auth-channel" enabled="true">
      <properties>
         <property name="httpAuthenticationChannelUri"
          value="https://backend.internal.example.com/auth"/>
      </properties>
   </provider>
</spi>
```

Here assumed that Decoupled Auth Server Reference Implementation run on https://backend.internal.example.com/auth

End