

**Document Version: <0.1>**

**OIDC & OAuth2 Implementation**

Contents

[1 Document Control 3](#_Toc512276061)

[1.1 Change Record 3](#_Toc512276062)

[2 PREFACE 4](#_Toc512276063)

[2.1 Project Overview / Context : 4](#_Toc512276064)

[2.2 Purpose of this document : 4](#_Toc512276065)

[3 OIDC Server’s Component(s) Summary : 5](#_Toc512276066)

[4 Component Logical Design & Different OIDC Flows / User-Journeys: 6](#_Toc512276067)

[4.1 High Level Component’s Logical Design 6](#_Toc512276068)

[4.2 Access-Token Flow 7](#_Toc512276069)

[4.2.1 Access-Token Flow (Client\_Credentials Grant Type) 7](#_Toc512276070)

[4.2.2 Access-Token Flow (Authorization\_Code Grant Type) 9](#_Toc512276071)

[4.3 OIDC + OAuth2 Authorization Journey: 11](#_Toc512276072)

[4.3.1 The roles end responsibility of Mule Gateway during the authorization journey : 12](#_Toc512276073)

[4.3.2 The roles end responsibility of Ping-Federate during the authorization journey : 13](#_Toc512276074)

[5 Ping – Federate Components & internal communication : 14](#_Toc512276075)

[6 Ping-Federate Service(s) Information : 15](#_Toc512276076)

[7 Ping-Federate Admin : 16](#_Toc512276077)

[8 Ping-Federate Run-Time : 17](#_Toc512276078)

[PingFederate clustering: 18](#_Toc512276079)

[9 Ping – Federate Data-Storage : 19](#_Toc512276080)

[9.1 10.1 In-Memory Data-Storage: 19](#_Toc512276081)

[9.2 External Data-Storage: 19](#_Toc512276082)

[9.3 Data-Store Management : 21](#_Toc512276083)

[10 Integration with External Login & Consent Applications : 22](#_Toc512276084)

[10.1 Ping-Federate Agentless Integration Kit: 22](#_Toc512276085)

[10.1.1 ReferenceId Adapter: 22](#_Toc512276086)

[10.1.2 Implementation Approach: 23](#_Toc512276087)

# Document Control

|  |  |
| --- | --- |
| **Title** | OIDC & OAuth2 Implementation |
| **Original Author(s)** | Nisarg Doshi – Capgemini |

## Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| **Version No.** | **Date** | **Author(s)** | **Revision Notes** |
| 0.1 | April-18th-2018 | Nisarg Doshi - Capgemini | Initial documentation |

# PREFACE

## Project Overview / Context :

The banking service is being radically transformed to promote innovation. This transformation is being driven by industry and regulatory directives – Open Banking APIs – UK CMA. As per this directives the Third Party Service Providers will consume standard banking APIS to provide the Account Information / Balance / Transaction & to provide the Payment Initiation relates services to the Bank customers. TPPs will access the customer’s bank accounts details through APIs and these APIs must be secured by OIDC standards and OB Security Profile. For this particular purpose , it becomes essential to adopt an security model underpinned by Open Banking Security Standards – OIDC & OAuth2.

Ping-Federate is a leading solution that can achieve compliance with the Open-Banking Security profile. Ping-Federate is a capable OAuth2 Authorization Server and OpenID Provider and meets the need of OB and PSD2 requirements. It can achieve following areas :

1. Integration with Bank’s login and Consent Applications
2. Integration with TPP portal for client registration / onboarding purpose to the Bank’s API Platform.
3. Integration with Ping-Directory (LDAP) for persisting the OAuth Clients and Grants (Tokens)
4. Integration with MuleSoft API Gateway to achieve secure access over external facing APIs.

## Purpose of this document :

This document provides details about design approach for using Ping-Federate as OIDC as well as OAuth Server in Capgemini API Platform. This design document talks about the high-level view of all the components which are part of OIDC Implementation as well as the details about actual implementation done at Ping-Federate Server. We as a Cap product team has used Ping-Federate Product to make our API Platform comply with OIDC & OB Security Profile Specifications.

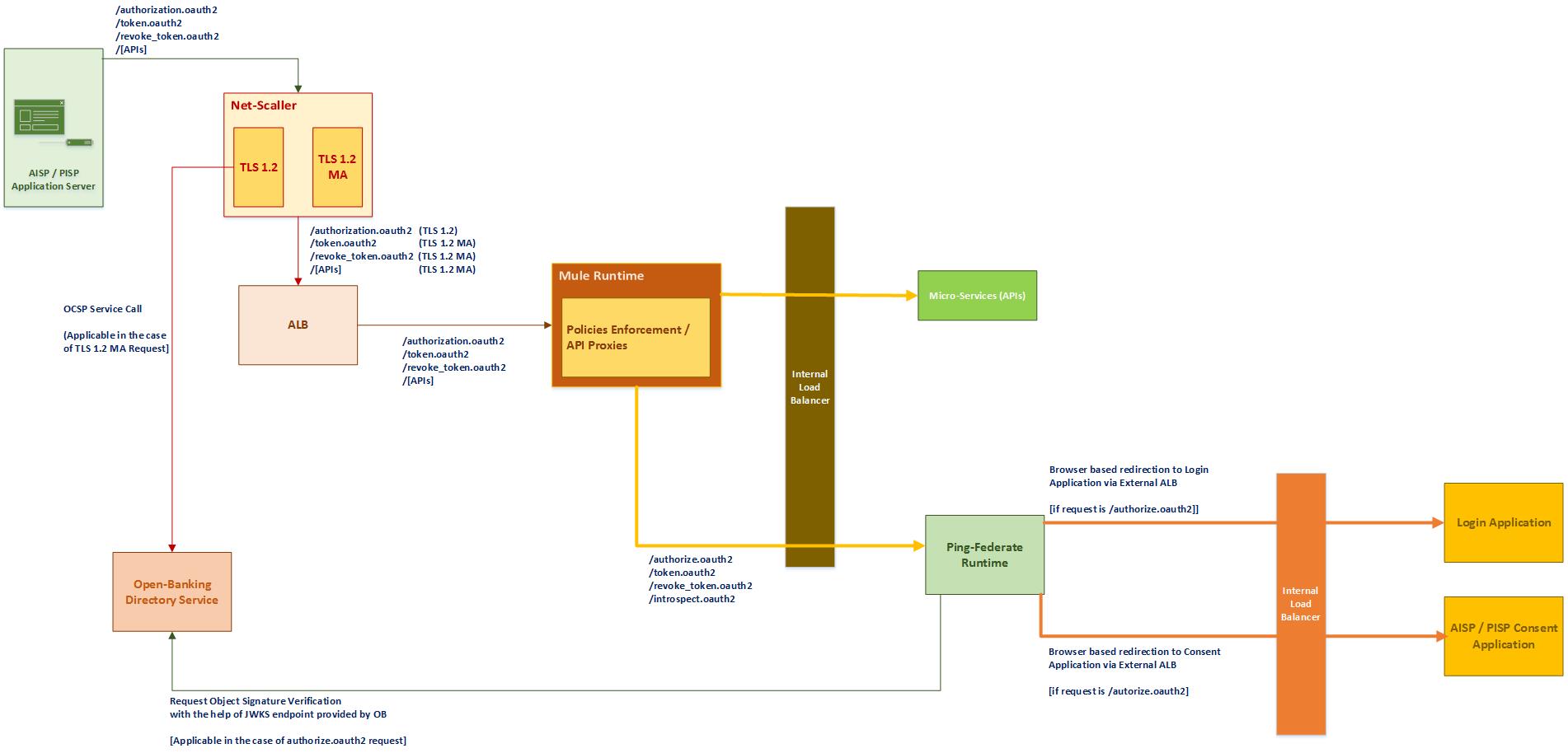
# OIDC Server’s Component(s) Summary :

|  |  |  |
| --- | --- | --- |
| **Component** | **Detailed Role** | **Role** |
| Mule | Mule Any point Platform acts as a API Gateway , it enforces the policies over the APIs before allowing the call to reach to downstream API layer. | API Gateway , OAuth2 Resource Server |
| Ping-Federate | Ping-Federate acts as an Open-ID Provider as well as OAuth2 server. It comprises with Ping-Federate Runtime & Ping-Federate Admin | OAuth2 & Open ID Connect Provider |
| Authentication Application | The angular js and spring boot based application which enables the customer to perform the authentication with Bank’s underlying systems. | IDP Application |
| Consent Application | The angular js and spring boot based application which enables the customer to authorize or reject the TPP’s request for accessing the accounts or payments services on the behalf of the customer. |  |
| Ping Directory | It provides the LDAP hierarchical data store for persisting the OAuth2 clients and grants (Tokens) as well as TPPs information. | Identity Storage |
| TPP Software / Application | Its OB registered client application which interacts with Platform APIs on the behalf of the cusromer | OAuth2 Client |

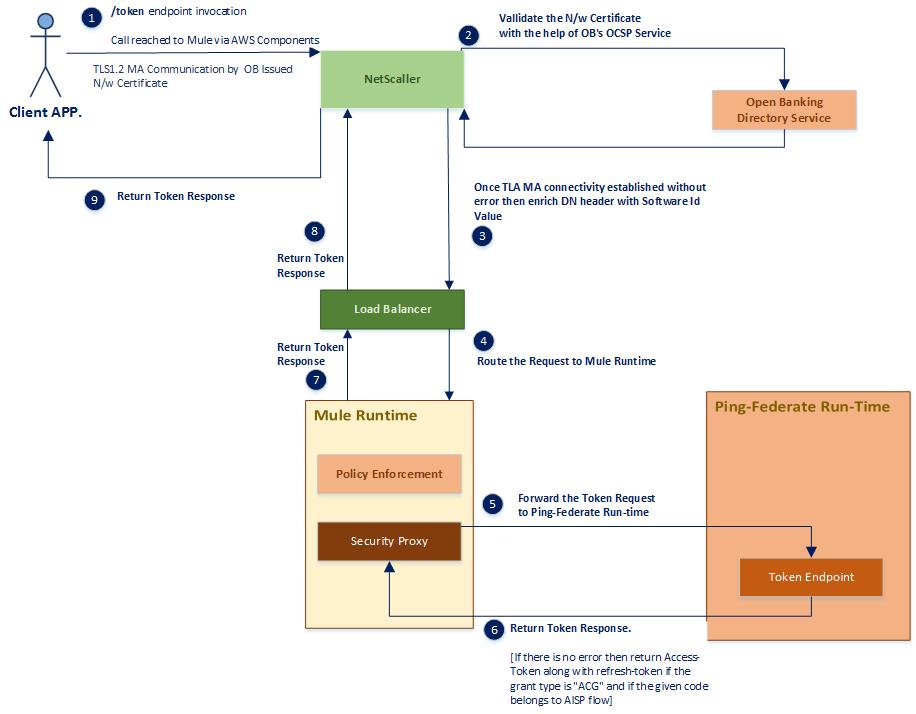
# Component Logical Design & Different OIDC Flows / User-Journeys:

In this section , we will provide details about components interaction in the context of all the OIDC & API Flows like Requesting Access-Token , Authorization Flow , Invoking AISP / PISP API.

## High Level Component’s Logical Design



## Access-Token Flow



As per OAuth2 protocol , TPP Software makes the token request by providing grant type. There are different grant types can be provided in the token request.

For each of the grant type , Token request flow behaves differently to met the requirement of Capgemini PSD2 API Platform and Open-Banking Security Profile Specification & OAuth2 protocol.

Please find detailed functional aspects associated with each of the grant type flow.

### Access-Token Flow (Client\_Credentials Grant Type)

The client application makes the token request with client\_credentials grant type in-order to get access-token to invoke account setup or payment setup APIs. This is the first step to be executed by client application before initiating the authorization journey. This is the POST request which will be required to involve as per the **client-secret-basic** mechanism. In the request payload client application has to provide mandatory field of grant\_type with the value as “**client\_credentials**” & Authorization Header with the Base64 encoded form of client id and secret as per Basic Authentication mechanism.

In-order to invoke the account setup API , client application needs access token with “accounts” scope and similarly to invoke payment setup API , client application needs access token with “payments” scope. Hence this request can be made with optional scope field as a value “accounts” or “payments” or “accounts payments” and this scope will be supplied as a part of post payload body. The current implementation is in accordance with **OAuth RFC 6749**.

#### The roles end responsibility of Mule Gateway in the client credentials token flow :

Following are the steps performed by Mule Any Point platform when the token request received with client\_credentials grant type.

**Following steps are actually done by set of custom policies :**

1. Perform Rate-Limiting on the basis of SLA Tier Enabled as well as the incoming CID and Secret present in the Basic Authorization Header.
2. Apply x-fapi-interaction-id policy which takes care of generating random x-fapi-interaction-id if its not provided in the request by client.
3. Validate incoming SSL Certificate DN Header with the incoming CID , both should match with each other.
4. Validate the client block status with the getting help of Platform API.

**Following steps are actually done by actual security mule proxy (If all above steps done without any error prone conditions) :**

1. If all these above constraints have been cleared then the pass on this request to the next layer component which is Ping-Federate Run-Time cluster via Internal Load-Balancer (ALB).
2. Return the Token Response to the client application / software (Caller) , it can be error response or actual response with opaque token.

#### The roles end responsibility of Ping-Federate Runtime Cluster in the client credentials token flow :

Following are the steps performed by Ping-Federate Runtime cluster when the token request received with client\_credentials grant type.

1. Validate the incoming request by performing some OOTB checks for example –
2. CID , Secret Validations
3. Grant Type Validation to check whether client is allowed to initiate token request with client\_credentials grant type or not.
4. Scope validation and this check will only take place if the scopes are actually provided in the request. This validation checks whether the requested set of the scopes are actually allowable to the caller client application or not.
5. If all the OOTB validations are executed without any error prone condition then populate and persist the token introspection storage for the new access token.
6. Return the opaque access token (or error response if any) to the calling layer which is Mule.

### Access-Token Flow (Authorization\_Code Grant Type)

The client application makes the token request with authorization\_code grant type in-order to get access-token / **refresh\_token (if applicable)** to invoke account balance / info / transaction , payment submission APIs. This step will be done by client application once the client application receives the auth\_code at the end of finishing the authorization journey succcessfully.

This is the POST request which will be required to involve as per the **client-secret-basic** mechanism. In the request payload client application has to provide mandatory field as mentioned below :

1. “**grant\_type**” with the value as “**authorization\_code**”
2. “**Authorization**” Header with the Base64 encoded form of client id and secret as per Basic Authentication mechanism.
3. **“code”** with the value as auth code which was given by Ping-Federate Server to client application at the end of authorization journey.
4. “**redirect\_uri**” with the value as the same redirect uri used by client application at the time of initiating the authorization journey.

The current implementation is in accordance with **OAuth RFC 6749**.

#### The roles end responsibility of Mule Gateway in the authorization\_code token flow :

Following are the steps performed by Mule Any Point platform when the token request received with authorization\_code grant type.

**Following steps are actually done by set of custom policies :**

1. Perform Rate-Limiting on the basis of SLA Tier Enabled as well as the incoming CID and Secret present in the Basic Authorization Header.
2. Apply x-fapi-interaction-id policy which takes care of generating random x-fapi-interaction-id if its not provided in the request by client.
3. Validate incoming SSL Certificate DN Header with the incoming CID , both should match with each other.
4. Validate the client block status with the getting help of Platform API.

**Following steps are actually done by actual security mule proxy (If all above steps done without any error prone conditions) :**

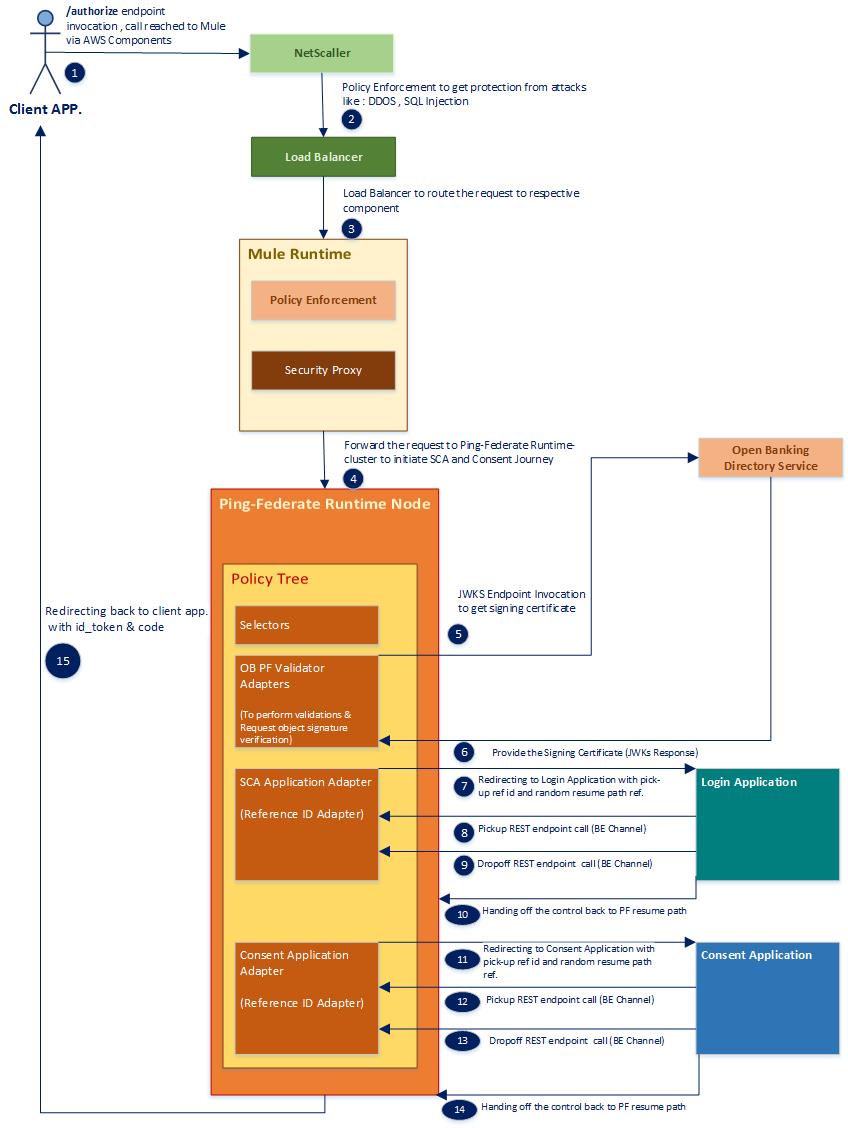
1. If all these above constraints have been cleared then the pass on this request to the next layer component which is Ping-Federate Run-Time cluster via Internal Load-Balancer (ALB).
2. Invoke the platform API to update the status of the intent & consent with “Authorized” status.
3. Invoke the PF Token introspection API with the actual token (received from Ping-Federate as a outcome of token request) to evaluate the scope present.
4. If the scope belongs to payments scope then exclude the the refresh\_token field from the HTTP JSON response .
5. Return the final Token Response to the client application / software (Caller) , it can be error response or actual response with opaque tokens.

#### The roles end responsibility of Ping-Federate Runtime Cluster in the authorization\_code token flow :

Following are the steps performed by Ping-Federate Runtime cluster when the token request received with authorization\_code grant type.

1. Validate the incoming request by performing some OOTB checks for example –
2. CID , Secret Validations
3. Grant Type Validation to check whether client is allowed to initiate token request with authorization\_code grant type or not.
4. Perform OOTP auth code validations with the help of incoming CID and Secret & redirect\_uri.
5. If all the OOTB validations are executed without any error prone condition then populate and persist the token introspection storage for the new access token.
6. Return the opaque access token (or error response if any) to the calling layer which is Mule.

## OIDC + OAuth2 Authorization Journey:



This authorization request has to be made in-accordance with the OIDC , OAuth2 Specifications as well as the Open-Banking Security Profile. As per the Open-Banking Security Profile the request has to be made according to the Hybrid flow with multiple response types in the request as “code idToken” and the request must be comply with OIDC + OAuth Specifications. In-order to initiate the combine journey of OIDC and OAuth , request must contains following things :

1. Scope : Must contains the value as “openid”
2. Response Type: Must be having two response types “code id\_token”
3. Request Object:

Request object must be provided with signed JWT. The request object must be provided as per OIDC Specification.

1. Redirect\_uri: Must be provided as valid value of client application’s registered uri.
2. client\_id: Must be provided as a valid value of registered software id.

### The roles end responsibility of Mule Gateway during the authorization journey :

When the authorization request comes to Mule-Platform , Mule platform acts as an Passthrough proxy for such browser based flow. In-addition to the passthrough proxy , it also performs below security enforcements before request actually reach to the Ping-Federate Runtime cluster :

**Following steps are actually done by set of custom policies :**

1. Apply global throttling through SLA Tier and this policy validates the incoming request against the specified threshold limit.
2. Apply x-fapi-interaction-id policy which takes care of generating random x-fapi-interaction-id if its not provided in the request by client.

**Following steps are actually done by actual security mule proxy (If all above steps done without any error prone conditions) :**

1. If all these above constraints have been cleared then the pass on this request to the next layer component which is Ping-Federate Run-Time cluster via Internal Load-Balancer (ALB).

### The roles end responsibility of Ping-Federate during the authorization journey :

Once the request reached to Ping-Federate Run-Time cluster to process the authorization request then Ping-Federate actually start performing all the **OOTB** OAuth +OIDC as well as the Ping-Federate Specific security validations. Once all these OOTB validations are completed without any error prone conditions then Ping-Federate’s policy tree will be executed . Actually this policy tree is chain of all the selectors and adapters which has been designed to met product and Open-Banking Security profile requirements. The roles and responsibility of this policy tree is listed out as below :

1. Validate the scopes combinations with the help of custom selectors. The allowed scopes must be having combination of “openid” , “accounts” or “openid” , “payments”.
2. Validate the Redirect URI query parameter with the redirect uri present in the request object.
3. Perform the TPP block check event with the integration of platform supporting API
4. Perform the Intent Validation with the integration of platform supporting API
5. Invoke the login application’s ReferenceID Adapter to launch the login application and to allow the authentication application to perform communication with Ping-Federate runtime via Back-End channel.
6. Takes the control back once the login application hand-back the flow back to Ping-Federate RunTime via user-agent.
7. If the hand-back was done from the login application for the purpose of terminating the user journey then redirect back to client software’s redirect uri with corresponding error code and error messasge.
8. If there is no such error condition involved as mentioned in the step 7 then return the approved auth code and id\_token to the client software’s redirect uri via user-agent.

# Ping – Federate Components & internal communication :

|  |  |
| --- | --- |
| **Components** | **Description** |
| Ping-Federate Admin Node | Ping-Federate Admin nodes push out changes to the all Ping-Federate Runtime nodes |
| Ping-Federate Runtime Node | Each run-time node sends and receives from each other node in the cluster. |
| Ping Directory | Ping-Federate connects to LDAPv3 data store for persistence of OAuth clients and grants |

# Ping-Federate Service(s) Information :

The following table summarizes the ports and protocols that PingFederate service(s) uses to communicate with external components.

**Note:**

Direction refers to the direction of the initial requests relative to PingFederate. Inbound refers to requests received by PingFederate from external components. Outbound refers to requests sent by PingFederate to external components.

|  |  |  |  |
| --- | --- | --- | --- |
| **Service** | **Protocol,**  **Direction,**  **Transport,**  **Default Port** | **Calling Source** | **Description** |
| Administrative Console | HTTPs  Inbound  TCP , 9999 | Administrative Engine | Used for incoming requests to the administrative  console. Configurable in the run.properties file. |
| Runtime Engine | HTTPS,  Inbound,  TCP, 9031 | Mule Runtime,  Login Application,  Consent Application | Used for incoming requests to the endpoints of authorize, token , introspection , pickup and dropoff. |
| Directory Service | LDAPS,  Inbound,  TCP 636 | Ping-Federate | Used for persisting the OAuth Clients and Grants. |
| PingFederate  Cluster traffic  (TCP) | JGroups,  inbound,  TCP, 7600 | PingFederate peer servers in a clustered PingFederate  environment | Used for communications between engine nodes  in a cluster when the transport mode for cluster  traffic is set to TCP (the default behavior). |
| PingFederate  Cluster traffic  (TCP) | JGroups,  inbound,  TCP, 7700 | PingFederate peer servers in a clustered PingFederate  Environment | Used by other nodes in the cluster as part of the  cluster's failure-detection mechanism when the  transport mode for cluster traffic is set to TCP (the  default behavior). |

# Ping-Federate Admin :

Following are the role of the Ping-Federate Admin :

1. The role of the PF Admin is to perform administrative tasks like doing configuration around Identity Provided , OAuth Server , System Configuration.
2. PingFederate Administrative API exposed by the PingFederate Admin node is a REST-based interface that provides a programmatic way to make configuration changes to PingFederate as an alternative to using the administrative console – this is available from https://<admin-cluster-dnsname>:<admin\_port>/pf-admin/api-docs.
3. It connects with all the runtime clusters available at system to get them in sync with each other.
4. The role of PF Admin is to push out the changes to the all Ping-Federate runtime nodes.

# Ping-Federate Run-Time :

Following are the role of the Ping-Federate Run-Time :

1. It provides below OAuth endpoints for external clients (TPPs) and they are proxied via Mule :

a. /authorize.oauth2

b. /token.oauth2

d. /revoke\_token.oauth2

1. It provides below endpoints for internal HOST to HOST communication purpose:
2. /introspect.oauth2
3. /pickup
4. /dropoff
5. Changes are always pushed in-to Run-time node via Admin console or API only.

# PingFederate clustering:

Please understand below bulleted points about clustering part :

1. Configuration changes are pushed out from the PingFederate admin node.
2. Runtime Engine nodes do not rely on the Admin node to operate and service requests.
3. Changes are pushed out via admin console or API only.
4. When a PingFederate node comes up it joins the cluster and checks with the Admin node to determine other nodes in the cluster, checks current configuration (PingFederate does a pull on initialization only).
5. Normal operation is that all runtime nodes interface with each other node.
6. PingFederate has a discovery list and makes a TCP connection (IP config) part of run.properties - iterates through the list until it finds the first one and then joins the cluster (multi-master).
7. PingFederate makes use of Jgroups for clustering - supports dynamic clustering.
8. Restarting of PF runtime nodes rarely required.
9. The run.properties file dictates the PingFederate role (pf.operational.mode) used and each node should be configured for single purpose operate as a runtime engine node or an admin node – never both.
10. PF cluster state is maintained by the runtimes.
11. State servers are runtime nodes not visible to the load balancer - dictated by the preferred node indices in the PingFederate run.properties file..
12. State management - each node send and receives to each other node in the cluster.
13. PingFederate Admin node must be a single node for a given cluster .
14. There will be no DR admin node and in the event that the PingFederate Admin node fails, the server build automation tooling will automatically build a new DR admin node. There can only be one node in the role of clustered\_console per PingFederate cluster.

# Ping – Federate Data-Storage :

## 10.1 In-Memory Data-Storage:

Ping-Federate uses a HyperSQL relational Java database that runs on PingFederate runtime engine nodes and is managed as black-box by the PingFederate runtime engine itself. It is known for its small size, ability to execute completely or partly in memory, its flexibility and speed. PingFederate uses this for user-session state and configuration data that are shared among the runtime servers enabling them to process requests as a single entity

## External Data-Storage:

The oauth client data attributes and meta data will be externalised by PingFederate into a new branch of an existing Directory Information Tree. ou=clients,ou=application,dc=boi,dc=co.uk,dc=capgeminibank,dc=com

For PingFederate v9.0, the oauth client attributes and metadata are covered in by a LDAP schema provided by the product which is applied to PingDirectory. The image below shows the DIT and an example oauth client instance.

##### PingFederate will interface to a PingDirectory data store with LDAP user credentials across LDAPs port 636.

##### The LDAP schema from Ping Identity creates a structural object class called pf-oauth-client that is used to store all the PingFederate OAuth 2.0 Client Information with following fields:

1. pf-oauth-client-name
2. pf-oauth-client-refresh-rolling
3. pf-oauth-client-logo-url
4. pf-oauth-client-hashed-secret
5. pf-oauth-client-description
6. pf-oauth-client-persistent-grant-exp-type
7. pf-oauth-client-persistent-grant-exp-time
8. pf-oauth-client-persistent-grant-exp-time-unit
9. pf-oauth-client-bypass-approval-page
10. pf-oauth-client-grant-type
11. pf-oauth-client-redirect-uri
12. pf-oauth-client-restrict-scopes
13. pf-oauth-client-restricted-scopes
14. pf-oauth-client-logout-uri
15. pf-oauth-client-authn-type
16. pf-oauth-client-client-cert-issuer-dn
17. pf-oauth-client-client-cert-subject-dn
18. pf-oauth-client-jwks-url
19. pf-oauth-client-jwks
20. pf-oauth-client-enforce-replay-prevention
21. pf-oauth-client-require-signed-requests
22. pf-oauth-client-supplemental-info
23. pf-oauth-client-last-modified

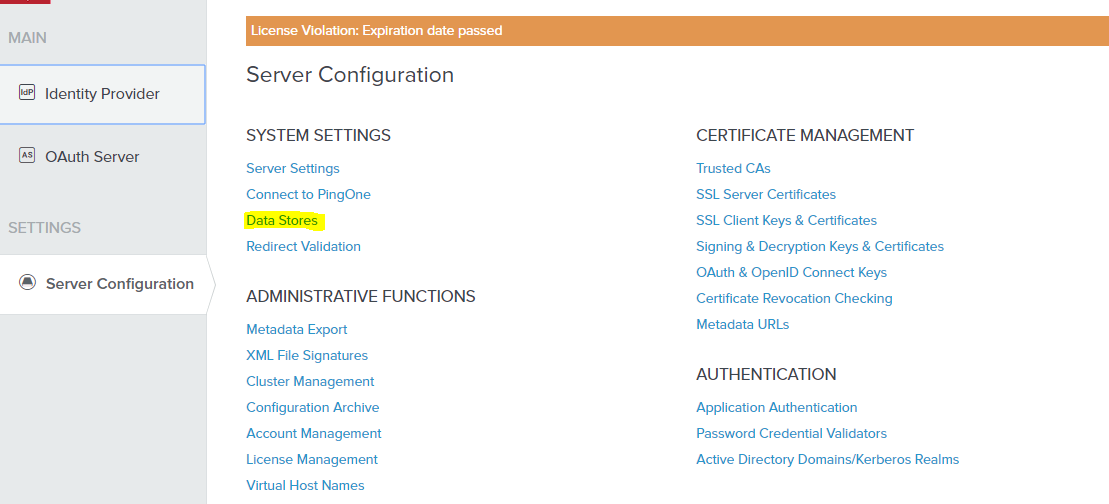
##### 

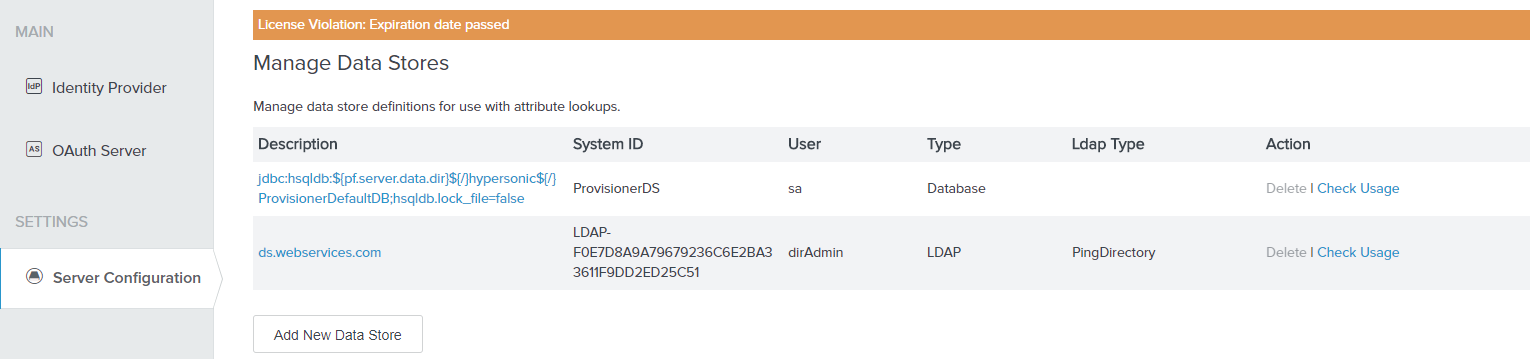
In addition access grants are persisted as follows under the different ou=AccessGrant under the same ou=application but a differ:

##### 

## Data-Store Management :

All data-stores can be managed by Admin Console , there a option of “Data Stores” available under SYSTEM SETTINGS which is a part of Server Configuration section :





# Integration with External Login & Consent Applications :

As a part of authorization journey , there are some steps involved through which user will perform authentication action with Bank’s underling host systems as well as user will take a action to authorize the TPP’s consent request. This journey takes place via user-agent of the customer (PSU) and it starts from the TPP’s web based application & takes the user-agent with different applications – login and consent applications .This journey ends with the user’s consent submission or rejection action which is a part of consent application. This authorization journey is actually involving TPP Application , User-Agent , OIDC Provider , OAuth2 Server , Login & Consent Applications. In the context of this project, Ping-Federate acts as an OIDC Provider as well as OAuth2 Server which requires the integration with two external applications – Login and Consent in-order to redirect the customer’s user-agent to login and consent applications (In sequential manner).

## Ping-Federate Agentless Integration Kit:

The Ping-Federate Agentless Integration Kit includes the ReferenceId Adapter which enables the external applications to integrate with Ping-Federate server. This integration can be achieved only if the applications will be integrated with Ping-Federate Server as an Identity Provider (IdP) or a Service Provider (SP). In the context of this project and the Capgemini API Platform, the Authentication Application will act as an Identity Provider and Ping-Federate will delegate the customer authentication responsibility to external authentication application.

The benefit of this adapter is that it’s a standard off-the-shelf adapter supplied by PingFederate and it follows a lightweight integration whereby the Identity Provider does not need to deploy any agent software in-order to interface with Ping-Federate, in fact the attributes are passed via HTTPs calls b/w external application and Ping-Federate through a “back-end” channel and server based interface.

### ReferenceId Adapter:

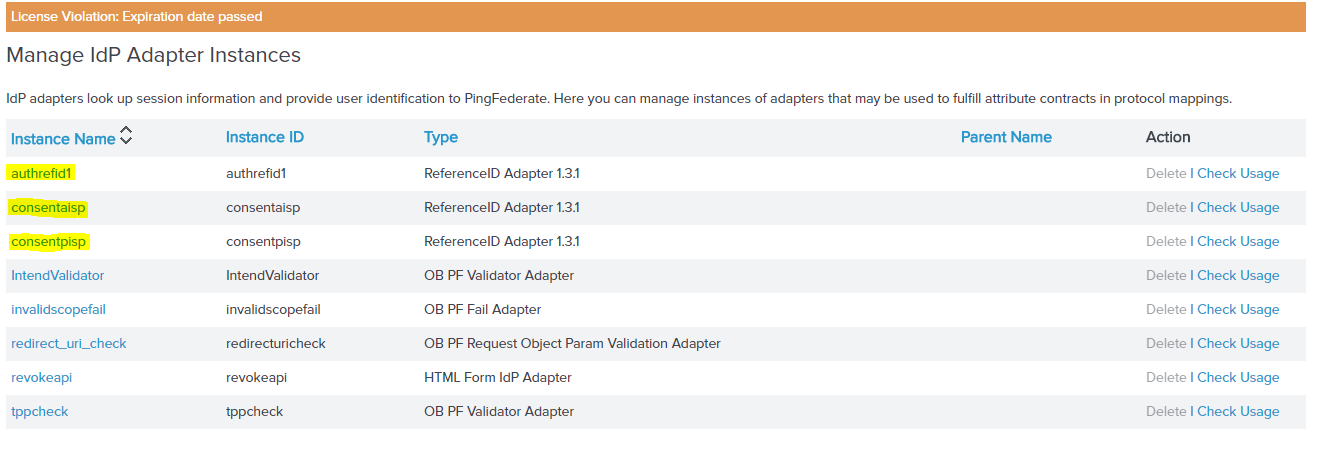
Ping-Federate Agentless Integration Kit includes the ReferenceId Adapter which allows the developer to integrate other external applications with Ping-Federate Server. In the context of our project and Capgemini API Platform, login and consent are two external applications which will be integrated with Ping-Federate Server by using this ReferenceId Adapter.

### Implementation Approach:

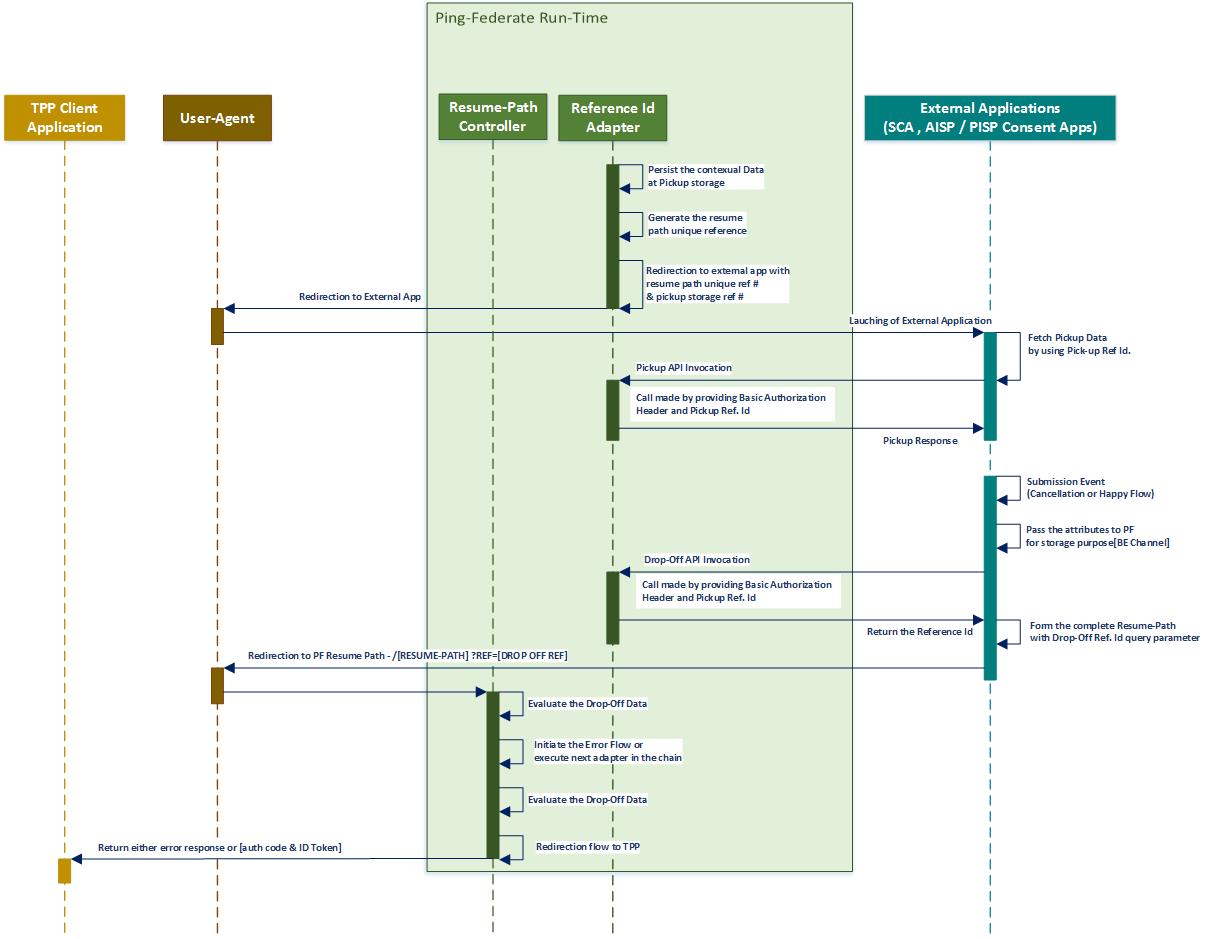
The Agentless Integration Kit ReferenceID Adapter JAR file can be downloaded from Ping Identity web site. The Adapter instances can be created on Manage IdP Adapter Instances configuration page

The following instances are created for the Open Banking project

1. authrefid1 - To integrate with Authentication (Signin) application
2. consentaisp - To integrate with AISP Consent application
3. consentpisp - To integrate with PISP Consent application



#### Flow b/w Ping-Federate & SCA / Consent Applications:



#### Authentication Application Reference Id Adapter:

##### **The sequence of invocation of Authentication ReferenceId Adapter :**

This adapter is a part of policy tree. Once all the OOTB pre-authentication & custom validations like intent id & tpp block check will be executed without any validation failure error then after this Authentication ReferenceId Adapter (named as a “authrefid1”) will be executed within policy tree.

##### **The role of Authentication ReferenceId Adapter :**

The role of Authentication ReferenceId Adapter is to generate unique pickup reference & unique call-back reference and launch the external authentication application with these two references and enables that external authentication application to connect with Ping-Federate runtime instance’s pickup & drop-off API interfaces to exchange the required attributes via “Back-End channel”.

#### AISP and PISP Consent Application Reference Id Adapters:

##### **The sequence of invocation of AISP or PISP Consent ReferenceId Adapter :**

There are two separate ReferenceId Adapters for AISP and PISP Consent Applications. Either of AISP or PISP Consent Application ReferenceId Adapter will be invoked within policy tree based upon the current value of OAuth Scopes supplied in the request. Out of these two adapters , one of the consent adapter will be executed in the policy tree (Chain) once the authentication process is done and control comes back to Ping-Federate without any error.

##### **The selection of the AISP or PISP Consent Adapter logic :**

The decision to launch either AISP or PISP Consent Application is always happens on the basis of the value supplied for OAuth scope in the authorization request. Below are the conditions :

1. If the value supplied for OAuth scope contains “openid” and “accounts” then AISP Consent ReferenceID Adapter will be invoked.
2. If the value supplied for OAuth scope contains “openid” and “payments” then PISP Consent ReferenceID Adapter will be invoked.

##### **The role of “Selectors” to solve the conditions :**

In-order to select the either of these two ReferenceId Adapters , we need to implement the “Authentication Selectors” and place it in the policy tree.

This selectors will compare its defined scopes with the scopes currently present in the OAuth + OpenId request. When such Selectors are configured on Policy Tree then it will fulfil the purpose of either invoking the AISP ReferenceID Adapter or PISP ReferenceID Adapter based upon the current scopes supplied in the request.

We actually need two below “Authentication Selectors” :

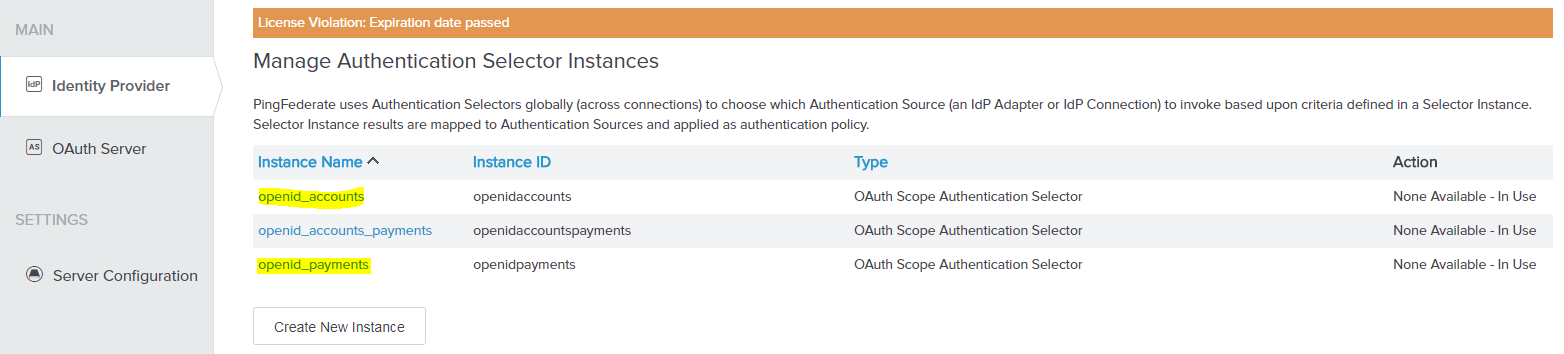
1. “openid\_accounts” :

It will be evaluated true if the scoped supplied in the request contains the combination of “openid” & “accounts”.

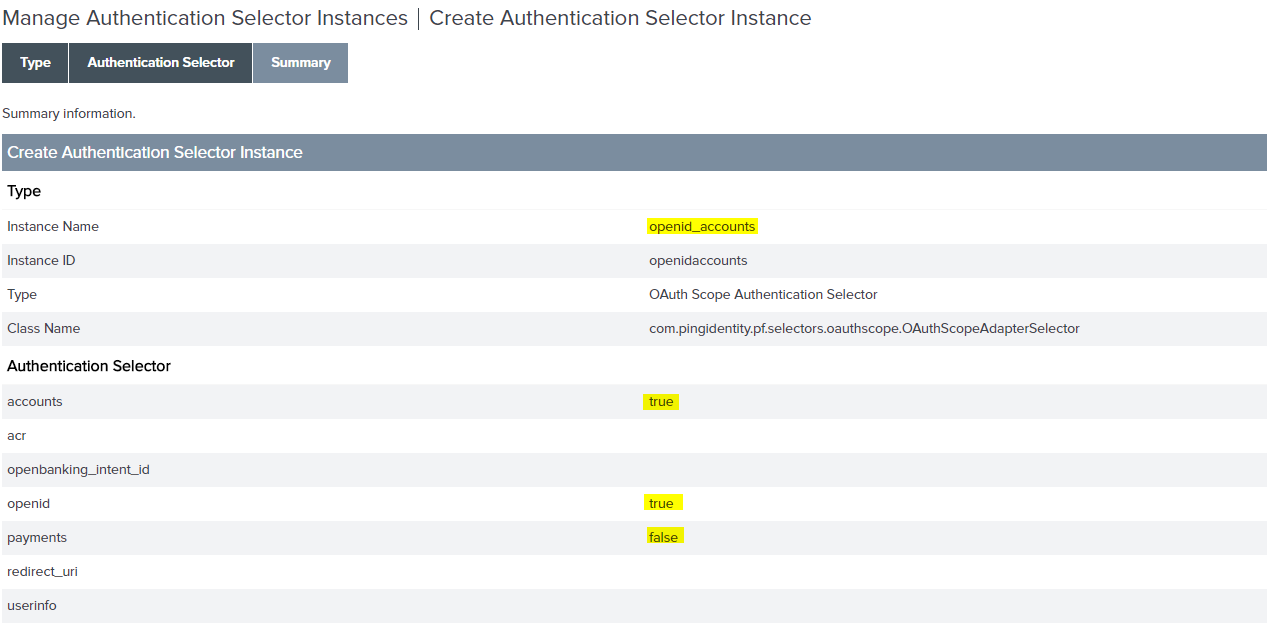
1. “openid\_payments” :

It will be evaluated true if the scoped supplied in the request contains the combination of “openid” & “payments”.

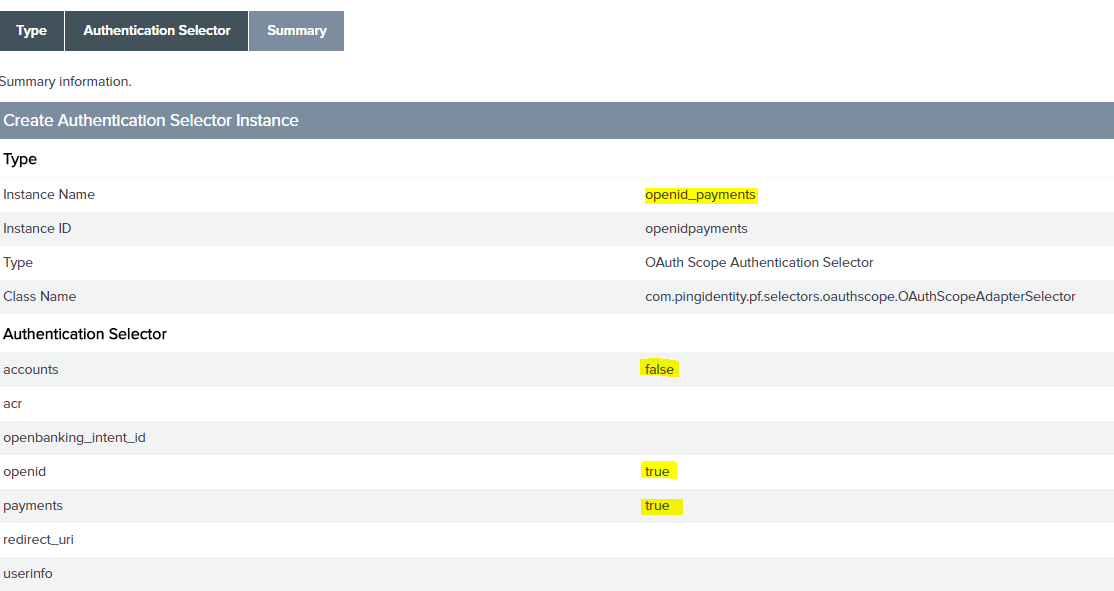
Refer below view of these Selectors :



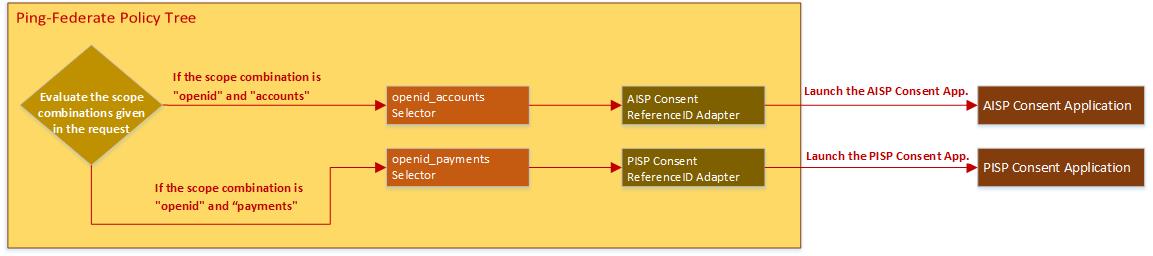
openid\_accounts selector properties :



openid\_payments selector properties :



##### **Conditional flow to invoke AISP or PISP Consent Application:**



##### **The role of AISP / PISP Consent ReferenceId Adapter :**

The role of Consent ReferenceId Adapter is to generate unique pickup reference & unique call-back reference and launch the external consent application with these two references and enables that external consent application to connect with Ping-Federate runtime instance’s pickup & drop-off API interfaces to exchange the required attributes via “Back-End channel”.

#### More Details about Reference ID Value :

The reference value is a long hexadecimal String. Length is determined by the Reference Length setting in the ReferenceID Adapter configuration. The default is 30 bytes.

Example: “A9C020F7CF8C21002CDC774B48A7CFE6B3ECA5FC6CCA507EE419B4432DB”

The reference value is short-lived (the default is three seconds) as specified by the Reference Duration setting in the ReferenceID Adapter configuration – this is also configured but should always be short lived.

The reference value is specific to the instance of the ReferenceID Adapter that issued it. If the ReferenceID Adapter is used both for authentication and consent application integration there are two distinct reference values and the values issued by the instance used for authentication are not valid for consent and vice versa..

The reference value is used only one time to prevent replay attacks. If the specified reference value is bad, the ReferenceID adapter returns an empty set of attributes.

In addition to Basic authentication, applications may use client-certificate authentication to communicate with PingFederate and the ReferenceID Adapter.

