**Proposal  
for MuleSoft Anypoint Platform  
security and access management  
detailed design**

**Author :** Tamas Torok  
**Reviewed:** Mario Orgaz

1. References

HLD 

OAuth 2.0 flows: [https://auth0.com](https://auth0.com/)

MulseSoft <https://docs.mulesoft.com/access-management/conf-client-mgmt-pf-task>  
<https://docs.mulesoft.com/api-manager/2.x/policy-ping-federate>  
<https://docs.mulesoft.com/access-management/external-identity>

1. Scope
   1. Security

This document adds information to Chapter 7.3. Security 1st point:

* 1. Access Management

This document seeks to add a concrete solution to Chapter 5.3. Access Management of the high-level design document (HLD) as Identitiy Management (IM) and Client Management (CM).

In case of IM, the solution covers the HLD 4.3.1. MuleSoft system level Integrations SSO chapter too.

The recommended Permission Model remains unchanged.

Deloitte recommends the MOL’s available PingFederate (PF) system for IM-CM role in case of external and user interactive access management as a single point of truth.

PF is an application supported by Mulesoft that

* can be seamlessly integrated with MOL’s AD/ADD IM systems and
* can provide SSO requirements according all integration channels and
* will fully support federated authentication of possible 3rd parties’ IMs too.

1. Out of scope

Message security for confidentiality (en/decrypting), integrity (signing) is not part of the default design. Optional the the Anypoint Platfrom can hold this functionality with addinal modules or external providers can also be integrated.

1. Network Security

On the Anypoint Platform (AP) for internal APIs they can only connect with other AP APIs, the inbound connections network security not necessary on HTTPS protocol. In this case, the caller network zone filtering and/or the caller address whitelist is a must.

It means the fine-tuning of the AP, relief instead of unnecessary overheads.

1. Identity Management

Dynamic Client Registration over OpenID Connect isn’t released yet with PingFederate provider.

* 1. Human identities

There are 3 cases for storing and managing: the human users and customers and Anypoint Platform users identity.

In this perspective, the **internal users** mean all enterprise employee who can access the internal resources. Their identites are gathering in AD/AAD.

**External customers** mean all registered and external client who can access the priviliged resources only via Anypoint Platform from public internet.

**Platform users** mean the all of internal=external users who can access the Anypoint Platform resources.

All of these groups (internal users, external customers, platform users) identiites can be stored as shared between more id provider.

In the proposed solution, the internal users are placed in AD/AAD that is hooked to PF and PF itself could hold the registered external customers. The platforms users are part of these two group above. Human user registration is manual for all three cases.

* 1. Application identities

AP APIs applications identification based on Anypoint Exchange support that generate client ID and secret for them (per organization/environment). This is the primary store for AP APIs identity. In case when System APIs should authenticate on external APIs with OAuth/OpenID Connect, the registration in external auth provider (PF) is also necessary.

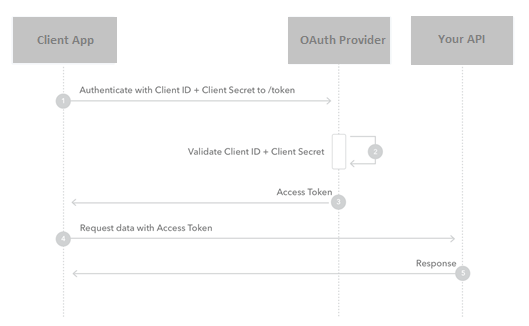
Any other external caller application sould be registered in PF external auth provider.

1. Client Management
   1. OpenID Connect\*\* on Experience APIs and AP management

The followed OAuth 2.0 terminology

* + Resource Owner: the entity that can grant access to a protected resource. Typically this is the end-user (internal user, external customer, platform user).
  + Client: an application requesting access to a protected resource on behalf of the Resource Owner.
  + Resource Server: the server hosting the protected resources. This is the API you want to access.
  + Authorization Server: the server that authenticates the Resource Owner and issues Access Tokens after getting proper authorization. In this case, PF.
  + User Agent: the agent used by the Resource Owner to interact with the Client, for example a browser or a native application.
    1. Client Credentials Grant

In the case of machine-to-machine authorization, the Client is also the Resource Owner, so no end-user authorization is needed.



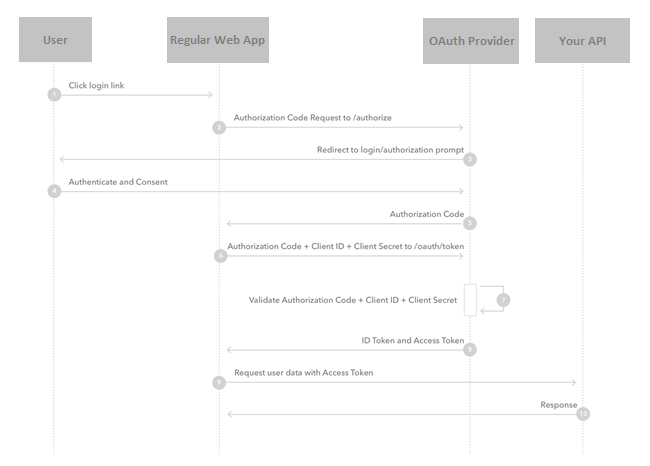
* + 1. Authorization Code Grant

If the Client is a regular web app executing on a server, then that is the flow you should use. Using this the Client can retrieve an Access Token and, optionally, a Refresh Token. It's considered the safest choice since the Access Token is passed directly to the web server hosting the Client, without going through the user's web browser and risk exposure.

If the Client is a Single Page App or native/mobile app, an application running in a browser using a scripting language like JavaScript, there are two grant options: the **Authorization Code Grant using Proof Key for Code Exchange (PKCE)** and the Implicit Grant. For most cases, we recommend using the Authorization Code Grant with PKCE.

Main steps

1. Get the authorization with applications ClientID and get back the Authorization Code from PF
2. Exchange the authorization code for an Access Token at PF
3. Call the API with access token
4. API will verify the token at PF



* + 1. Resource Owner Password Credentials Grant

In this flow, the end-user is asked to fill in credentials (username/password), typically using an interactive form. This information is sent to the backend and from there to oauth provider. It is therefore imperative that the Client is absolutely trusted with this information. This grant should only be used when redirect-based flows (like the Authorization Code Flow) are not possible.

* 1. ClientID Enforcement\*\*\*\* on Process and System API

The Client ID Enforcement policy enforces the requirement for credentials. Rate Limiting - SLA-Based and Throttling - SLA-Based policies use the client ID as a reference to impose limits on the number of requests that each application can make within a period of time.

This recommendation follows the HLD 7.3. Security chapter 2nd point.

* 1. For access the external systems from AP, several auth methods are supported on System API.

In case of direct DB connection (only in necessary cases) DB technical user credential authentication or Kerberos is recommended. In this ways, the credential or Kerberos oriented files should be encryted in AP secure configuration properties.

In case of OAuth or SAML based authentication, the System APIs should have registration in external auth provider (PF).

1. Katie’s story   
   (use case for hooked and federated access management)



1. Katie signs in to her iPhone *Customer Mobile App (App).*

2. Part of the digital solution catering to Katie’s experience is the *My Shopping API* which delivers all the relevant capabilities to her iPhone *App*. This API is protected with the **PingFederate access token enforcement policy \***. In order to consume the *My Shopping API*, the iPhone *App* must interact with the *PingFederate Authorization Server* to request an **OpenID Connect** \*\* token on her behalf. PingFederate authenticates her credentials against Active Directory. Upon successful authentication, it generates a token and signs it (**Digital Signatures** \*\*\*) before responding to the iPhone *App* with the token. The *App* presents this token in a custom HTTP header on every subsequent call to *My Shopping API*.

3. All calls between *experience, process and system APIs* are protected with the **client ID enforcement policy \*\*\*\***. Each API has an ID and secret stored in **Mule Credentials Vault \*\*\*\*\***.

4. **Optional message security for special case as confidentiality, reliability:** The calls to the Tokenization API are protected with **client cert authentication** and the payload passed to the API is encrypted and signed with **Mule Encryption Processor and Mule Digital Signature Processor** respectively before sending.

5. **MQ is a special case for JMS only:** **Publications to Anypoint MQ** are protected with OAuth 2.0 and HTTPS.

6**. MQ is a special case for JMS only: Subscriptions to Anypoint MQ** are protected with OAuth 2.0 and HTTPS.

7. There is a federated trust between the Identity Providers in both security contexts for Mythical Retail and its shipping partner. Both the sales associate’s *App (Clienteling App, Customer Portal)* and Katie’s *App* can call the *Order Tracking API* with the access token that they received from PingFederate. The *Order Tracking API* validates the token with OpenAM, the Identity Provider of the shipping company. This is able to verify that the token was signed by the Identity Provider of Mythical Retail, which it trusts. The *Order Tracking API* accepts the invocation by the iPhone *App* and responds accordingly.

8. **Optional MFA:** Sales associates use multi--‐factor authentication to sign in to their *App.* The *App* calls *PingFederate* OAuth 2.0 authorization server to get a token which it passes to the *Shop Assistant API*.

9. Interaction with the systems of record is secured in various forms according to the requirements of each *Server (back-end apps).* Tokens and username and password credentials are stored in **Mule Credentials Vault \*\*\*\*\*.**

\* External access token enforcement policy

It validates incoming tokens previously issued by PingFederate or Open AM OAuth Provider upon receipt of client ID and secret.

\*\* OpenID Connect with JWT ID Tokens

Tokens OpenID Connect is built on top of OAuth 2.0 to provide a Federated Identity mechanism that allows you to secure your API in a way similar to what you would get were you to exploit WS-­‐Security with SAML. It was designed to support native and mobile apps while also catering for the enterprise federation cases. It is an attractive and much more lightweight approach to achieving SSO within the Enterprise than the corresponding WS-Security with SAML. Its simple JSON/REST based protocol has resulted in its accelerating adoption. Apart from OAuth 2.0 access tokens, OpenID Connect uses JWT (jot) ID tokens, which contain information about the authenticated User in a standardized format. Your API can make an access control decision by calling out to a UserInfo endpoint on the Identity Provider to verify if the User pertains to a certain role. Just like SAML Assertions, JWT ID tokens are digitally signed (see Digital Signatures 2.1.1) so a federated Identity Provider can decide to accept them based on its trust relationship with the Identity Provider that issued them.

\*\*\* Digital Signatures

We humans sign all kinds of documents when it matters in the civil, legal and even personal transactions in which we partake. It is a mechanism we use to record the authenticity of the transaction. The digital world mimics this with its use of Digital Signatures. The idea is for the App to produce a signature by using some algorithm and a secret code. Your API should apply the same algorithm with a secret code to produce its own signature and compare the incoming signature against this. If the two match, the API has effectively completed authentication by guaranteeing not only that this message was sent by a known App (only a known App could have produced a recognizable signature), but that it has maintained its integrity because it was not modified by a third party while in transit. As an added benefit for when it matters with third party Apps, the mechanism also brings non-­‐repudiation into the equation because neither the App, nor the User can claim not to have sent the signed message.

\*\*\*\* Client ID enforcement policy

Locks down your API for consumption only by a set of known clients.

\*\*\*\*\* Mule Credentials Vault was module below 4.X version

Credentials Vault is for the encryption of properties which are referred to and decrypted by the Mule application at deployment time. These properties are encrypted with a variety of algorithms and are completely hidden from anyone who does not have the key to the credentials vault. At deployment time, the key is passed to Mule as a system property. This key should only be in the hands of authorized personnel.

Credentials Vault exchanged to Secure Configuration Properties at version 4.X.