Capgemini API Platform



OpenID Provider - Detailed Solution Design

|  |
| --- |
| **Date: 27th March 2018** |
| **Version: v1.0** |
| **Classification: Confidential** |

Document Control

|  |  |
| --- | --- |
| Title | OpenID Provider – Detailed Solution Design |
| Author | Barry O’Donohoe |

|  |  |
| --- | --- |
| Owner (Responsible for Approval of Issued Versions) | |
| Name | Role |
|  | TBC |

|  |
| --- |
| Review Panel |

|  |  |  |
| --- | --- | --- |
| Name | Role | Review Requirement |
| Bhupinder Saini | Design & Engineering Lead | Mandatory |
| Daryl Searle | Scrum Master / Project Manager | Mandatory |
| Gareth Coffey | Ping CI/CD Technical Lead | Mandatory |
| Rishabh Shah | Capgemini Product Owner |  |
| Gary Lehane | BOI Security Architect | Mandatory |
| John O’Connor | BOI Enterprise Business Architect | Mandatory |
| Dheepson Brightson | BOI Enterprise Architect | Mandatory |
| Ramandeep Singh | Capgemini Technical Lead | Mandatory |
| Nisarg Doshi | Capgemini Manager FS GBU | Mandatory |
| Rajesh A. Shetty |  | Information |
|  |  | Information |
|  |  | Author Requested |
|  |  | Author Requested |

|  |  |  |  |
| --- | --- | --- | --- |
| Change History | | | |
| Issue | Date | Author/ Editor | Details of Change |
| Issue 1 Draft A | 24th Nov 2017 | Barry O’Donohoe | Early draft for internal review |
| Issue 1 Draft B | 1st Dec 2017 | Barry O’Donohoe | Second early draft not for full review |
| Issue 1 Draft C | 15th Dec 2017 | Barry O’Donohoe | 3rd draft following receipt of feedback from draft B and new sections added. |
| Issue 1 Draft D | 5th February | Barry O’Donohoe | 4th draft following receipt of feedback from draft B and new sections added. |
| Issue 1 Draft E | 23th March | Barry O’Donohoe | 5th and final draft following receipt of feedback from earlier draft. |

TABLE OF CONTENTS

1 Preface 4

1.1 Purpose of this document 4

1.2 Objectives and Scope 4

1.3 Project overview / context 5

1.4 Project scope 6

2 System Overview 7

2.1 System Characteristics 7

2.2 System Architecture 8

2.3 Technology Stack 9

2.4 Component summary 9

2.5 Logical component design 10

2.6 Integration Connectivity 11

2.7 Ports & Protocols 13

2.8 Data Architecture 13

2.9 System & MI requirements 18

2.10 Security Design 18

2.11 Cryptographic Key Inventory 21

3 Integration design – Authentication & consent 22

3.1 Authentication & Consent Component Interaction Diagram 23

3.2 Consent & Authentication Sequence Flow Diagram 24

3.3 Pingfederate agentless integration kit 24

4 System Design 27

4.1 URL Naming conventions 27

4.2 Design Decisions 27

5 Component Description 28

5.1 State management design 28

5.2 PingFederate clustering 28

5.3 Ping deployment topology and decisions. 29

6 system configuration 30

6.1 Mulesoft Token Provider 30

6.2 PingFederate DataSTORES 30

6.3 Listeners & Security Settings (Certificates & Key Pairs) 30

6.4 Access token management 30

6.5 Oauth Client (various) 30

7 Appendix 31

7.1 PIngfederate endpoints 31

8 Definitions, Acronyms and Abbreviations 36

8.1 PSD2 Glossary 36

8.2 References 36

# Preface

## Purpose of this document

This document is a Detailed Solution Design Document produced by external consultancy RAIDIAM for its clients, Capgemini and the Bank of Ireland. This design specifies detailed component level information covering commercial software components sourced from Ping Identity and its direct integrations to sub-system components.

This design shall be used to inform detailed infrastructure design specification and associated installation and configuration tasks in the target hosting environment. It shall be updated as necessary during the implementation phase to take account of any changes such that it becomes the definitive system reference point for all detailed component design information related to the Ping Identity component and its direct integrations.

## Objectives and Scope

This document shall describe the detailed design for PingFederate acting as an OpenID Provider (OP) as part of the Capgemini Platform to achieve compliance with the Open Banking Security Profile and related standards. In addition it will address the compliance to the Open Banking Security Profile and the Financial API (FAPI) conformance standards on which it is based. It describes in a complete and consistent way how the physical (deployable) components interact to realise the requirements including a complete description of the physical elements that will be deployed into the live environment for the component.

**The design is comprised of six viewpoints as indicated in table below**

| Viewpoint | Description |
| --- | --- |
| Functional | The physical functional elements, their responsibilities, the interfaces they expose, and the interactions between them. This viewpoint demonstrates how the Sub-system actually performs the functions required of it. |
| Information | How the Sub-system will store, manipulate, manage, and distribute information. |
| Concurrency | How the Sub-system handles any required concurrency and state-related structure and constraints. The parts of the Sub-system that are required to run at the same time and how this is controlled will be included in this viewpoint through the use of process and state models. |
| Development | Describes the build and configuration management of deliverables, system-wide design constraints, and system-wide standards to ensure technical integrity of the Sub-system. |
| Deployment | Defines the logical deployment environment in which the Sub-system is intended to run, including the hardware environment the Sub-system needs (e.g., processing nodes, network interconnections, and disk storage facilities), the technical environment requirements for each node (or node type) in the Sub-system, and the mapping of software elements to the runtime environment that will execute them. |
| Operational. | Defines how the Sub-system can be controlled, managed and monitored. This viewpoint will fundamentally take into account the desired operating vision of the Capgemini API Platform team. |

**Table 1: Viewpoint Descriptions**

## Project overview / context

The banking services landscape is being radically transformed to promote increased competition and innovation. This transformation is being driven by industry and regulatory directives that envisions Open Banking APIs – UK CMA & EBA (*European Banking Authority Payment Services Directive (PSD)2 and Competition & Markets Authority Open Banking*). Third Party Providers (TPPs) will consume standard banking APIs to provide Account Information and Payment Initiation Services to customers of Banks’. TPP access to accounts (XS2A) must be secured by Banks’ using Strong Customer Authentication (SCA) per technical standard, RTS (Regulatory Technical Standard). With traditional security perimeters dissolving, a new approach is needed to ensure security postures remain within risk appetite. Enabling this vision necessitates an identity-centric security model underpinned by open international standards - OAuth2 & OIDC.

Bank of Ireland (BOI) as a CMA 9 bank operating in the UK have engaged Capgemini to deliver an API Platform that achieves compliance with the PSD2 and Open Banking(OB) mandates. The API Platform is currently geared towards an API security model based upon OAuth2; and of late it became known that OB would align to OIDC (OpenID Connect). The API Platform must be enhanced to achieve compliance with the Open Banking standard - the existing ‘Auth server’ does not support OIDC. PingFederate was already proposed to deliver the federation needs of the API platform and the obvious first choice to investigate. PingFederate is a leading solution that can achieve compliance with the Open Banking security profile and has been selected to replace the ‘Auth server’ component.

PingFederate is a capable OAuth2 Authorisation Server and OpenID Provider and an excellent choice to meet the OB and PSD2 requirements. This will be achieved by:

* Deploying PingFederate to meet all federation and SSO needs of TPPs, developers and BOI staff members.
* Integrating with the Bank authentication and consent applications - both as external concerns.
* Integration with the TPP Developer Portal for registration/onboarding to the Bank API Platform.
* Integration with existing PingDirectory instance for persistence of OAuth clients and persistent grants.
* Integration with the Mulesoft API Gateway to secure all external REST API endpoints.

In addition, Bank of Ireland have the opportunity to:

1. Establish a modern set of IAM capabilities that are standards based, responsive to changing needs, emerging threats and vulnerabilities.
2. Deliver a common set of adaptive, risk aware and context-based authentication and authorisation services across Web, Mobile & API.
3. Achieve a seamless and secure customer user experience that maximises conversion and retention rates with reduced high-friction controls
4. Secure regulatory compliance with industry directives whilst maximising potential in a PSD2 era of Open Banking APIs.
5. Enable the delivery of an omni-channel experience across web, mobile and API using new digital framework underpinned by the API Platform.

## Project scope

The primary aim of the project is to swap out a major and central component of the Capgemini API Platform architecture in order to obtain compliance with the Open Banking standard. PingFederate has been identified and must be introduced into the stack in the most elegant and least disruptive fashion to maximise reuse of existing components whilst minimising rework.

The scope includes:

* Delivering a PingFederate cluster into an AWS VPC environment to act as an OpenID Provider (OP) and OAuth2 Authorisation Server (AS).
* Integrating with an existing PingDirectory backing store for oauth client and persistent grant storage.
* Integrate with the Mulesoft API Gateway which acts as the OAuth resource server.
* Integrate with the Bank authentication and consent applications –both as external concerns.
* Integrate with the Capgemini TPP Developer Portal to support onboarding and registration
* Integrate with the OBIE Directory component, in particular JWKS endpoints to obtain TPP public keys to verify the signature on the OIDC authorisation request object.
* Integration to Splunk for logging and monitoring purposes.
* Exposing JWKS, Authorisation, Token, Introspection and well-known endpoints.
* test suite

The scope for this phase of the project and design excludes:

* TPP servicing apart from initial client creation during TPP registration/onboarding via Capgemini portal.
* Customer Consent Management portal – for a customer to self-manage (view/revoke) existing consents.
* Staff Consent Management Portal – for a staff member to manage existing customer consents on their behalf – assisted servicing through remote channels e.g. telephony.
* Customer Authentication or Consent Applications apart from their integration during authorisation grant.
* Any servicing requirements of TPPs (suspension) by BOI staff.
* Internal staff API access control.
* Client registration - Dynamic on-boarding – out of scope.
* Customer SSO from any of existing BOI web channels.
* No securing of any existing web channels.
* API security model for internal server to server communication - out of scope server to server API security.
* Integrations between API platform and on premise federation hub.
* Single authentication method - (PIN) unlocked by TouchID to access the native app for a soft-token - day 1.
* No expectation to implement a API security approach for consent, authentication.
* Integration to Fraudnet device profiling (JS snippets) - deal with response codes and passes to payment engine - logon and consent apps.
* MA-TLS - integrations to Netscaler
* OBIE integration except for TPP JWKS integration as part of OIDC request object processing.
* Full support for the final PSD2 RTS including embedded/decoupled authorization approaches.

# System Overview

Ping Identity is a US-based, mid-cap sized Identity and Access Management (IAM) product and service provider. Its products have been designed from the ground-up as a modern set of IAM capabilities based on open standards and APIs. The components are well conceived with good separation of concerns.

Capgemini/BOI have acquired licenses to host and operate a core product from Ping Identity which is:

1. PingFederate
2. An authentication gateway product to support all aspects of authentication for Web | Mobile | API
3. A capable identity provider that supports outbound and inbound identity federation and Single Sign-On use cases.
4. Underpinned by identity stores such as Active Directory for workers or strategic identity stores for customers.
5. Compliant with open international standards including SAML, WS-Federation, WS-Trust, OAuth, OpenID connect.
6. Extensive out of the box integrations with enterprise platforms, 3rd party IDPs & SaaS solutions.

PingFederate will operate in unison with the Mulesoft API Gateway to integrate identity-based access management policies using open standards access protocols. The Ping Identity component shall initially be deployed into a Capgemini virtual private cloud (VPC) provided in a public cloud hosting environment from AWS in the Ireland (eu-west) region across multiple availability zones (AZs). It will be deployed into several pre-production test environments followed by a production environment.

## System Characteristics

PingFederate is a lightweight component requiring a Java Runtime Environment (JRE) and will be hosted in this on a hardened Amazon Linux-derived AMI (Amazon Machine Image) using the Anitian PCI Hardened Amazon Linux virtual servers based on AWS EC2 instances.

PingFederate comprises runtime and administrative nodes. The runtime components support all user and client interactions to facilitate authentication and authorisation. The admin node is primarily providing an administrative web administrative console and REST API endpoints.

The initial delivery focus is for inbound (to Bank of Ireland) TPP interactions on behalf of customers to read/write account and payment API resources. The primary focus is to achieve compliance with the Open Banking security profile and related API standards using existing logon (authentication) and consent applications as part of the Capgemini API Platform.

In this phase, the BOI customer authentication application will be retained as-is utilising a mobile native application soft-token based on the HID Global ActivID solution. PingFederate will integrate with the authentication web application using the Agentless Integration Kit (also known as the Reference ID adapter) to achieve customer single sign-on during the authorisation grant. The authentication application is effectively acting as an identity providing application to PingFederate which trusts it has authenticated the user before creating a PingFederate authentication session.

To enable authorisation and access control, the Mulesoft API Gateway component will sit logically in front of API resource servers that it is protecting operating as a reverse proxy. All inbound requests to the resource server will be intercepted by Mulesoft so that access control policies can be enforced.

API requests received by Mulesoft will only be permitted to pass to the resource server if Mulesoft is satisfied that the user principal and optionally the initiating client is authenticated and authorised to consume the resource.

Ping Identity can support many other use cases including:

* 1. TPP Primary Technical Contact (PTC) inbound identity federation form Open Banking as an OP
  2. Customer identity federation through social login with the likes of Facebook, Google, etc.
  3. Worker outbound identity federation to SaaS offering such as servicenow, Office 365.
  4. Worker outbound identity federation to public cloud IaaS/PaaS services e.g. AWS, Azure.

Ping Identity provides a single trusted door way through which all access control is managed.





**Figure 1 – System Context – Ping Identity**

## System Architecture

PingFederate exposes several web and API endpoints to support open standards for authentication and authorisation including SAML2, WS-\*, OAuth, OpenID. PingFederate can support many types of end user authentication using knowledge based (single factor), hardware token (as a second factor), OTP and many others. Where that is the case, PingFederate can serve the web presentation logon pages – this is not applicable in this phase as the reference ID adapter being configured in PingFederate does not expose any web presentation elements which will be covered by the BOI authentication application directly.

PingFederate runtime nodes will use an external LDAP directory (PingDirectory) for persistence of OAuth client configuration including persistent OAuth grants for users. Splunk shall be used for all runtime log events of the PingFederate runtime components. PingFederate Admin shall log all events to a Splunk instance for the API Platform with a view to Splunk then forwarding to a Bank of Ireland SIEM – out of scope of this design.

## Technology Stack

The solution is will be made up of the following technology stack

* Java 1.8 SE runtime.
* PingFederate version 9.0.
* Host operating system – Anitian PCI Hardened Amazon Linux AMI.
* PingDirectory for data persistence of oauth client and persistent grants (refresh token).
* AWS Elastic Load Balancers (ELB/ALB) for reverse proxy, TLS termination and load distribution across available runtime nodes used internally within the AWS availability zones and not exposed externally to TPPs.
* Citrix Netscaler for Mutual TLS termination, reverse proxy

## Component summary

##### The table below provides a functional summary of the logical components:

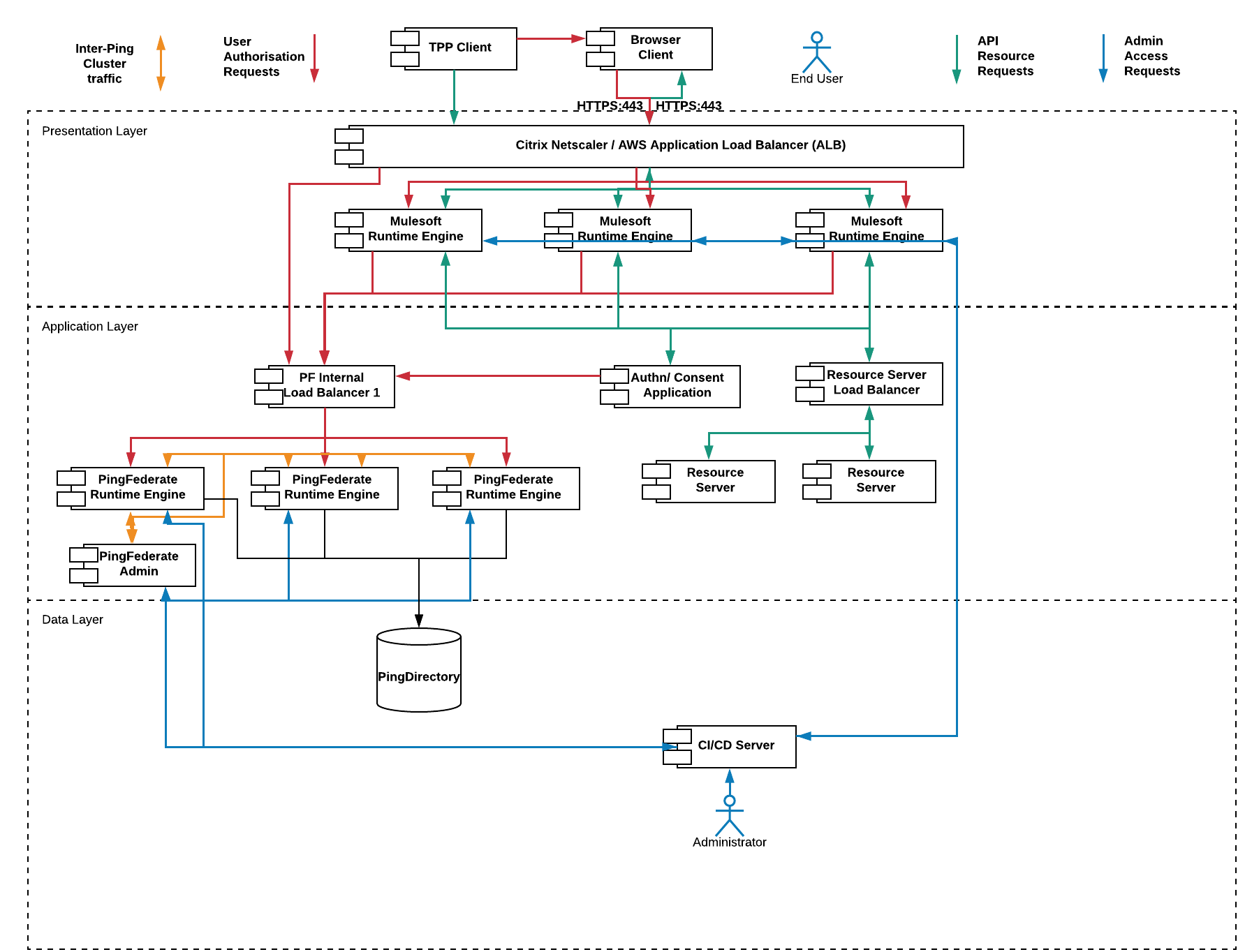
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Component** | **Description** | **New | Change** |
| **1** | IAM Policy Enforcement Point (PEP) | A PEP is a component that serves as a gatekeeper to a digital resource – it intercepts incoming requests and obtains a policy decision (Allow or Deny) from a Policy Decision Point (PDP) and enforces that decision e.g.  permit onward connection to resource. | Existing  (Mulesoft Anypoint) |
| **2** | IAM Identity Provider | A system that creates, maintains and manages identity information for principals (users, services, or systems) and provides principal authentication to other service providers within a federation or distributed network and send attribute assertions containing trusted information about the user.  Comprises:   1. PingFederate Runtime - The primary component supporting authentication, identity federation, Single sign-on and delegated authorisation from PingFederate Client browser; mobile devices; web service calls to the SSO Directory Service, the OAuth Client Management Service, and the OAuth Access Grant Management Service; Session Revocation API REST calls 2. PingFederate Admin - provides a Web Admin Console and REST API component used to administer all changes to the cluster and for support and maintenance of the cluster | New  (PingFederate) |
| **3** | Authentication Application | The angular js and springboot application that provides customer user authentication. | Existing |
| **4** | Consent Application | The angular js and springboot application that provides customer consent authorisation. | Existing |
| **5** | Identity Manager | A system to manage all identity classes in a holistic fashion throughout its lifecycle including it ABAC / RBAC entitlements. | Possible Future |
| **6** | IAM Identity Store | A central repository/persistent store for holding identity attributes and related metadata attributes including entitlements for (RBAC /ABAC).  PingDirectory - An LDAPv3 hierarchical data store for holding OAuth clients and persistent grants including TPP and Primary Technical Contact (PTC) metadata. | Existing  PingDirectory / Foundation Svc |
| **7** | Authentication Service | Foundation wrapper service to match customer user identifier and the corresponding OTP challenge/response in conjunction with HID Global appliance. | Existing – Foundation Service |
| **8** | Resource Server | The server hosting the protected resources that the IAM PEP is protecting. In OAuth2 terms the IAM PEP is performing the role of ‘resource server’. For the most part this is a web or API (RESTful micro service application). | Existing |
| **9** | TPPs | The Third-Party Provider software clients that interact on behalf of customers | New |
| **10** | OBIE | The Open Banking Implementation Entity and it’s Directory component | New |

## Logical component design

##### 

**Figure 2 – Solution Context**

The diagram below illustrates the components and their interactions at a logical level:



##### **Figure 3 – Ping Identity Component Design**

## Integration Connectivity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Call from | Call to | Protocol | Description |
| 1 | Browser | Load Balancer for PingFederate | HTTPS | Access to Authorisation endpoint to initiate authentication and consent as part of an authorisation grant |
| 2 | Load Balancer | Mulesoft API Gateway | HTTPS | Routing of API requests to the trusted control point for policy enforcement for all PingFederate and other protected read/write API endpoints |
| 3 | Mulesoft API Gateway | Load Balancer for PingFederate runtime engines | HTTPS | All inbound requests to PingFederate runtime engines from Mulesoft runtime engines originating from clients external to BOI. Mulesoft acting as a pass-through proxy. |
| 4 | Load Balancer for PingFederate runtime engines | PingFederate runtime engines | HTTPS | All inbound requests to PingFederate runtime engines from Mulesoft runtime engines originating from clients external to BOI. Mulesoft acting as a pass-through proxy |
| 5 | Authentication Application, Consent Application | Internal Load Balancer for PingFederate runtime secondary TLS port | HTTPS (TLS) | Authentication and Consent application will invoke the PingFederate agentless integration kit (reference ID adapter) through an internal load balancer (operating on a pass-through basis – not TLS terminating) in order to achieve single sign-on for authentication and consent. |
| 6 | Internal Load Balancer for PingFederate runtime secondary TLS port | PingFederate runtime secondary TLS port | HTTPS (TLS) | PingFederate runtime will terminate the incoming TLS connection from the authentication and consent applications using server-side TLS and basic authentication. The Load balancer is operating on a pass-through basis and not TLS terminating the connection from Authentication or Consent applications to PingFederate runtime secondary TLS Port. |
| 7 | Mulesoft API Gateway | PingFederate via Internal Load Balancer | HTTPS | OAuth, OpenID and other endpoints as required in performing role of OAuth client and resource server – this covers token introspection communications where Mulesoft speaks to a back-channel interface of PingFederate. |
| 8 | Mulesoft API Gateway | Resource Server Load Balancer | HTTPS | Mulesoft will act as a reverse proxy for all incoming requests and when satisfied that IAM policies have been enforced, permit access to the resource server and initiate a new TLS connection to the resource server load balancer |
| 9 | Resource Server Load Balancer | API Resource Server | HTTPS | The Resource Server load balancer will distribute incoming requests from Mulesoft across the API resource servers |
| 10 | PingFederate Admin nodes | PingFederate runtimes | Ping internal cluster comms | PingFederate Admin nodes push out changes to the all PingFederate runtime nodes. |
| 11 | PingFederate runtime nodes | PingFederate runtime nodes | Ping internal cluster comms | Each runtime nodes sends and receives from each other node in the cluster. |
| 16 | PingFederate Admin and Runtime nodes | PingDirectory | LDAPS Port 636 | PingFederate will connect to a LDAPv3 data store for OAuth client config and persistent grants. |

## Ports & Protocols

The following table summarizes the ports and protocols that PingFederate 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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Service | Protocol, Direction,  Transport, Default Port | Source | Destination | Description |
| 1 | Administrative Console | HTTPS, inbound, TCP, 9999 | Administrator browser, administrative API REST calls, web service calls  to the Connection  Management Service  Applicable to the console node in a clustered PingFederate  Environment | Administrative Engine | Used for incoming requests to the administrative  console. Configurable in the run.properties file. |
| 2 | Runtime Engine – primary port | HTTPS, inbound, TCP, 9031 | Client browser; mobile devices; the TPP oauth client and mulesoft runtimes.  Applicable to all runtime engine nodes  in a clustered PingFederate environment | Runtime Engine | Used for incoming connection to the various runtime endpoints including authorisation, token and introspection |
| 3 | Runtime Engine – secondary port | HTTPS, inbound, TCP, 9032 | Used for TLS authentication of the PingFederate runtimes – used for secured /dropoff or pickup of attributes using the referenceID adapter. Applicable to all runtime engine nodes  in a clustered PingFederate environment | Runtime Engine | Used for incoming connection to the various TLS runtime endpoints specifically for /dropoff and /pickup as part of referenceID adapter. |
| 4 | Directory Service | LDAPS,  outbound,  TCP 636 | PingFederate | PingDirectory | Used for communications between PingFederate  and PingDirectory for the purpose to storing and reading oauth client data |
| 5 | PingFederate  Cluster traffic  (TCP) | JGroups,  inbound,  TCP, 7600 | PingFederate peer servers in a clustered PingFederate  environment | PingFederate | Used for communications between engine nodes  in a cluster when the transport mode for cluster  traffic is set to TCP (the default behaviour). |
| 6 | PingFederate Cluster traffic  (TCP) | JGroups,  inbound,  TCP, 7700 | PingFederate peer servers in a clustered PingFederate  environment | PingFederate | 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 behaviour). |

## Data Architecture

|  |  |
| --- | --- |
|  |  |
| PingFederate Runtime Node | Runtime state is shared across runtime engines using embedded distributed Java database that acts as a black box of the PingFederate product stored locally on the file system of the host VM. |
| PingFederate – Admin Node | Configuration data and custom adapter data are stored locally on the file system of the host VM. There is also a cluster management set of data and also keys to validate cluster members. All of this data is stored on the local file system. |
| PingDirectory | PingDirectory stores OAuth client meta data and oauth persistent grant records. |

##### The PingFederate administrative console (CLUSTERED\_CONSOLE) failover process has been automated to provide a resilient solution, without any manual intervention required.

##### When a PingFederate cluster is deployed, a replica administrative console is created and is configured to join the cluster with its operational mode set to that of a runtime (CLUSTERED\_ENGINE) so that it can stay current with cluster configuration. This allows the replica to receive all cluster communications, and configuration, but it serves no active role in handling runtime requests.

##### A service runs on the replica that monitors the primary administrative console instance, and so it can become active should the single primary admin console node fail for any reason.

### Data Storage

The oauth client data attibutes 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

##### It is here that PingFederate will create/read/update and delete oauth clients created through the PingFederate Administrative Web console or REST API interface. 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.

##### 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

##### pf-oauth-client-name

##### pf-oauth-client-refresh-rolling

##### pf-oauth-client-logo-url

##### pf-oauth-client-hashed-secret

##### pf-oauth-client-description

##### pf-oauth-client-persistent-grant-exp-type

##### pf-oauth-client-persistent-grant-exp-time

##### pf-oauth-client-persistent-grant-exp-time-unit

##### pf-oauth-client-bypass-approval-page

##### pf-oauth-client-grant-type

##### pf-oauth-client-redirect-uri

##### pf-oauth-client-restrict-scopes

##### pf-oauth-client-restricted-scopes

##### pf-oauth-client-logout-uri

##### pf-oauth-client-authn-type

##### pf-oauth-client-client-cert-issuer-dn

##### pf-oauth-client-client-cert-subject-dn

##### pf-oauth-client-jwks-url

##### pf-oauth-client-jwks

##### pf-oauth-client-enforce-replay-prevention

##### pf-oauth-client-require-signed-requests

##### pf-oauth-client-supplemental-info

##### 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:

##### 

### Logging

##### PingFederate will generate these logs to capture server events:

##### admin.log - Records actions performed by administrative-console users.

##### admin-event-detail.log - Records detailed information about each applicable administrative-console event performed by administrative-console users if detailed event logging is enabled.

##### admin-api.log - Records actions performed by administrative-API users.

##### runtime-api.log - Records actions performed by API users using the OAuth Client Management Service, the OAuth Access Grant Management Service, and the Session Revocation API.

##### transaction.log - Records individual identity-federation runtime transactions at specified levels of detail.

##### audit.log - Records a selected, configurable subset of transaction log information plus additional details, intended for security-audit and regulatory compliance purposes.

##### provisioner-audit.log - Records Outbound Provisioning events, intended for security-audit purpose.

##### provisioner.log - Records only provisioning activity.

##### server.log - Records PingFederate runtime and administrative server activities.

##### init.log - Records only Jetty messages generated prior to PingFederate start up.

PingFederate uses the Log4j 2 logging service to generate its log files. Configurations are maintained in the log4j2.xml file, located in the<pf\_install>/pingfederate/server/default/conf directory.

**Log levels and verbosity**

Log messages are categorized into six log levels:

1. FATAL
2. ERROR
3. WARN
4. INFO
5. DEBUG
6. TRACE

Starting with version 8.2, PingFederate only records messages that are tagged with log level INFO, WARN, ERROR, and FATAL to the server log (and the provisioner log). Messages that are tagged DEBUG (or TRACE) are not recorded to optimize performance. Console logging is also disabled for the same reason.

For troubleshooting purpose, you may adjust the log level to DEBUG in the log4j2.xml file and optionally re-enable console logging.

**Fields (and attributes)**

The audit log and the administrative API log, can be customized to log additional (or less) information by modifying their pattern elements. Available fields are documented inline in the log4j2.xml file. In addition, PingFederate can be configured to log user attributes (if they are present) in the audit log, transaction log, and server log. When privacy is required for sensitive user attributes have their values masked in these logs.

HTTP requests to the runtime engine and the administrative console are logged to the *<date>*.request.log file and*<date>*.request2.log, respectively, by the Pingfederate web container. Like other PingFederate-generated log files, the HTTP request logs are written to the default PingFederate log directory. Properties controlling request logging are contained in the web-container configuration files:

* + jetty-runtime.xml for the runtime engine (the *<date>*.request.log files)
  + jetty-admin.xml for the administrative console (the *<date>*.request2.log files)

These configuration files are located in the <pf\_install>/pingfederate/etc directory and independently managed on a per-server basis.

#### API audit logging

PingFederate provides API endpoints and management services on the administrative port (9999) and the runtime port (9031). Actions performed through these endpoints are logged for auditing purposes, as described below:

| **API** | **Port** | **Log File** |
| --- | --- | --- |
| Administrative API | Administrative Port | admin-api.log |
| OAuth Client Management Service | Runtime Port | runtime-api.log |
| OAuth Access Grant Management Service | Runtime Port | runtime-api.log |
| Session Revocation API | Runtime Port | runtime-api.log |

Actions performed using the administrative API will be captured in log entries that include

* Time the event occurred on the PingFederate server
* Administrator username performing the action
* Authentication method
* Client IP
* HTTP method
* REST endpoint
* HTTP status code

Each of the above fields is separated by a vertical pipe (|) for ease of parsing.

#### Security Audit Logging

PingFederate® records a subset of transaction log information with additional details at runtime, intended to facilitate security auditing and regulatory compliance. PingFederate records this information in the audit.log file, located in the <pf\_install>/pingfederate/log directory. The following table describes the default elements that PingFederate writes to the audit log (in the order that they are listed):

| **Element** | **Description** |
| --- | --- |
| %d | Transaction time. |
| trackingid | The tracking ID values uniquely identify user sessions, useful for correlating log messages in the audit and server logs. |
| event | The type of transaction; for example, SSO, OAuth, AUTHN\_ATTEMPT, and AUTHN\_REQUEST.  AUTHN\_ATTEMPT and AUTHN\_REQUEST indicate an authentication attempt against an IdP adapter instance and an authentication request sent to another IdP partner (through an IdP connection), respectively. |
| subject | The subject of the transaction or authentication attempt. |
| ip | Incoming IP address. |
| app | The target SP application (when available). |
| connectionid | The connection identifier associated with the transaction. |
| protocol | The associated identity protocol; for example, SAML20 or OAuth20. |
| host | PingFederate host name or IP address. |
| role | The role of PingFederate played for the transaction. |
| status | The status of the transactions. |
| adapterid | The ID of an adapter instance.  Consider adding the **authenticationsourceid** and **targetsessionid** elements to log additional information about the request. |
| description | Description of an authentication failure (when information is available from an IdP adapter). |
| responsetime | Time elapsed (in milliseconds) from when a final request for a transaction is received to when the audit message is written. This value serves as an approximation of total transaction processing time and may be useful for monitoring trends. |

Each element is separated by a vertical pipe (|). Elements are configurable by editing the <pf\_install>/pingfederate/server/default/conf/log4j2.xml file.

The following table describes other elements (in alphabetical order) that can be added to the audit log:

| **Element** | **Description** |
| --- | --- |
| accessgrantguid | The GUID of the OAuth access grant (for OAuth transactions). |
| assertionid | The unique ID for the SAML assertion. |
| attrackingid | The tracking ID for OAuth access token. It could be used to analyze the flow of OAuth access tokens in the audit log and between PingFederate and PingAccess. |
| attributes | User attributes received (for an SP log) or sent (for an IdP log). |
| authenticationsourceid | An array of one or more IdP adapters or IdP connections (or both) invoked in an authentication or logout flow; for example, [idpAdapterOne, idpConnectionX] |
| granttype | OAuth grant type. |
| initiator | (SAML 2.0 only) The federation role that initiated the SSO or SLO: SP or IDP. |
| inmessagetype | Incoming message type. Possible values are Request or Response. |
| inresponseto | The value of the **InResponseTo** attribute of an SSO or SLO response. |
| inxmlmsg | The incoming message; for example, a SAML AuthnRequest or the information pertaining to an OAuth request. |
| localuserid | The local ID used for the transaction (when account linking is enabled at the SP). |
| outxmlmsg | The outgoing message; for example, a SAML Response or the information pertaining to a response for an OAuth request. |
| pfversion | The PingFederate version. |
| requestid | The ID of a SAML request. |
| responseid | The ID of a SAML response. |
| requeststarttime | The start time of the request in milliseconds since midnight, January 1, 1970 UTC. |
| stspluginid | For WS-Trust STS transactions, the ID for the token-processor or token-generator instance. |
| targetsessionid | An array of one or more SP adapters or SP connections (or both) invoked in an authentication or logout flow. |
| validatorid | The ID of the Password Credential Validator instance (for the successful attempts). |
| virtualserverid | The virtual server ID of a request (if applicable). |

##### The best practice guidance from Ping Identity is to use Splunk as the consolidation point for all events and management information from the system. They provide a Splunk App that will provide insight into the performance and utilisation of the solution.

## System & MI requirements

|  |  |
| --- | --- |
| 1. PingFederate – runtime node |  |
| Uptime, Status, Throughput, Errors, Informational and Warning information, administrative logon, change notifications, reboot notifications and cluster leave/join notifications from the application layer.  From the OS layer we will need Machine status, network status, memory status, last admin logon to the OS |
| PingFederate – Admin Node | Uptime, Status, Throughput, Errors, Informational and Warning information, administrative logon, change notifications, reboot notifications and cluster leave/join notifications from the application layer.  From the OS layer we will need Machine status, network status, memory status, last admin logon to the OS. |

## Security Design

Administrative controls: The Ping Identity solution stack will be subject to

a.    Privileged access controls and an elevated rights process to limit logical access to the administrative console of PingFederate as required under change and incident management circumstances.

b.    All administrative tasks in PingFederate will be undertaken by using named accounts that identify the actor responsible for the access which will be recorded and retained for detective purposes in audit log records.

c.     All access will be subject to role based access controls following the least privilege model such that the user has the minimum amount of access required to perform the activities required.

2.     Network Controls:

a.    AWS Virtual server hosts will be deployed across the AWS Ireland (eu-west) region AZs (1a-c) with PingFederate in the Application Zone – not required to be in an internet-facing DMZ. This separation of the host infrastructure will serve to isolate at a network level the various functional capabilities. All interconnections and integration points across the stack will be secured through firewall policy implemented with AWS security groups that observes the least privilege model – unidirectional, host IP based, no dynamic ranges, secure protocols etc..

b.    AWS Elastic and Application Load Balancers (VIPs) will be used to reverse proxy inbound access for IP termination in the originating zone and will use digital certificates for server authentication and transport encryption using strong ephemeral cipher suites (TLSv1.2) with perfect forward secrecy on outer Netscaler VIPs exposed to the internet.

* All certificates used on internal API Platform ALBs will be sourced from an internal self-signed Issuing CAs delivered by Capgemini.
* The authorisation external endpoint will use a certificate from a Commercial Public CA
* The Token and API external endpoints will use digital certificates obtained from an OpenBanking hosted and operated PKI as the trust anchor for the all ecosystem participants (ASPSPs and TPPs). HSTS (HTTP Strict Transport Security) will be enforced also on these VIPs.



c.    PingFederate will rely upon the common capabilities of the Capgemini API Platform for DDOS protection based upon Citrix Netscaler to ensure the internet presented Web and API endpoints are protected from any unexpected DDOS attacks. Specific protections are covered in the API Blueprint documentation.

* 1. In addition, the Citrix Netscaler will implement WAF controls to protect all PingFederate exposed endpoints exposed to the internet for customer authentication using standard signatures and black-listing of known bad source IPs, specifically:

|  |  |  |
| --- | --- | --- |
| **SNo.** | **Rule Name** | **NetScaler Features** |
| **1.** | Injection | Injection attack prevention (SQL or any other custom injections such as OS Command injection, XPath injection, and LDAP Injection), auto update signature feature. |
| **2.** | Broken Authentication and Session Management | AAA, Cookie Tampering protection, Cookie Proxying, Cookie Encryption, CSRF tagging, |
| **3.** | Cross Site Scripting (XSS) | XSS Attack Prevention, Blocks all OWASP XSS cheat sheet attacks. |
| **4.** | Insecure Direct Object References | StartURL checks, AAA, Form protections, and Cookie tampering protections. |
| **5.** | Security misconfiguration | PCI reports, SSL features, Signature generation from vulnerability scan reports such as Qualys. Additionally, very specific protections such as Cookie encryption, proxying, and tampering. |
| **6.** | Cross Site Request Forgery | CSRF form tagging, Referrer header validation. |

3.     Detective Controls:

a.    All system and audit events generated by the Ping Identity stack will be logged for monitoring purposes and will be forwarded to the Splunk system where a Ping Splunk App will provide a tailored view of specific metrics. The metrics will provide a lot of different operational dashboards covering SSO transactions, system health, service reports and trend analysis.

b.    APM solution to be used for application level monitoring with AWS Cloudwatch for infrastructure level monitoring used for monitoring the health and availability of the virtual servers hosting the Ping Identity stack to ensure the host is operating within normal thresholds of compute utilisation etc. It is anticipated that this will be based on the AppDynamics product.

4.     Security configuration:

a.    The Ping Identity solution will be configured to ensure all settings enforce best practice and least privilege for setting up OAuth2, Open ID Connect profiles, client IDs, client secrets and credentials, authorisation policies. Clients of the platform also have obligations to ensure they adopt best practice in their implementation of client code to mitigate any protocol weaknesses and undertake appropriate risk assessment and penetration testing.

b. Root and intermediary Certificate Authority (CA) digital certificates of the Capgemini internal enterprise PKI will be imported into PingFederate. This is important to establish trust anchors used for trusting digital certificates presented during secure HTTPS connections.

The CI build automation toolset will be used to generate all RSA key pairs and submit to various Capgemini issuing certificate authorities to obtain digital certificates used by PingFederate to secure client requests on the administrative console port and on the runtime engine port. All digital certificates used by the Ping Identity stack will use RSA 2048 bit key sizes with RSA2046Sha256 based signature algorithms. All digital certificates shall be imported into the PingFederate admin engines and will then be stored in a local trust store on each instance filesystem. Note: it is not intended to enable certificate revocation checking using a certificate revocation list distribution point or OCSP responder on PingFederate itself.

As an OpenID Provider, PingFederate supports the generation and rotation of temporary asymmetric key pairs to sign ID Tokens for Relying Parties - these in-memory short-term keys. In additional PingFederate can create and maintain server signing certificates, which you may use to sign outgoing requests, responses, assertions, and access tokens. The same type of certificate is also used for decryption.

From PingFederate version 9.0, the product does include the feature to permit an externally generated certificate to be used as the basis for digitally signing id\_tokens – this is critical to achieve full compliance with the Open Banking Security Profile standard. One of the challenges with the implementation of the new feature in PingFederate is that the KID (Key ID) value does not agree with that derived from the OBIE directory. Therefore, the FAPI conformance suite (or any client for that matter) cannot identity and obtain the correct public key from the OB hosted JWKS endpoint as the OP issued id\_token KID in the id\_token JWS (JSON Web Signature) header is a different value.

This has been conveyed to and confirmed by PingFederate Product Owner and will be fixed in version 9.0.3 expected in the next couple of months (tentative May 2018). This will mean a partial implementation only in the meantime whereby the PingFederate as OpenID provider will digitally sign the id\_token using an OB Directory issued certificate, but the PingFederate hosted JWKS will need to be used instead of the OB hosted one – this is of no significance to the TPP client or end user and will be transparent.

It does mean that an OB initiated or managed certificate revocation would necessitate the PingFederate key to be updated so that the JWKS reflects the correct status (ordinarily the OB hosted JWKS would be automatically updated to remove the revoked public key from the JWKS endpoint).

## Cryptographic Key Inventory

| **Component Name** | **Key Name** | **Purpose of Cryptography** | **Public Key Signer (CA)** | **Private Key Location** | **Key Owner** | **Key Support Team** |
| --- | --- | --- | --- | --- | --- | --- |
| PingFederate | ADMIN (1 x Certificate & Key), ENGINE (1 x Certificate & Key) | Server authentication &  Protection of data in flight | Capgemini Internal Enterprise PKI | Private Keys will be centrally generated by the CI build automation tooling and pushed to the server where they will be held in local Java Key Store based trust store protected by filesystem controls. | <<Service Owner TBC>> | <<App Support team TBC>> |

Note: All certificates will use 2048 bit RSA key sizes with SHA-256 based signature algorithms.

# 

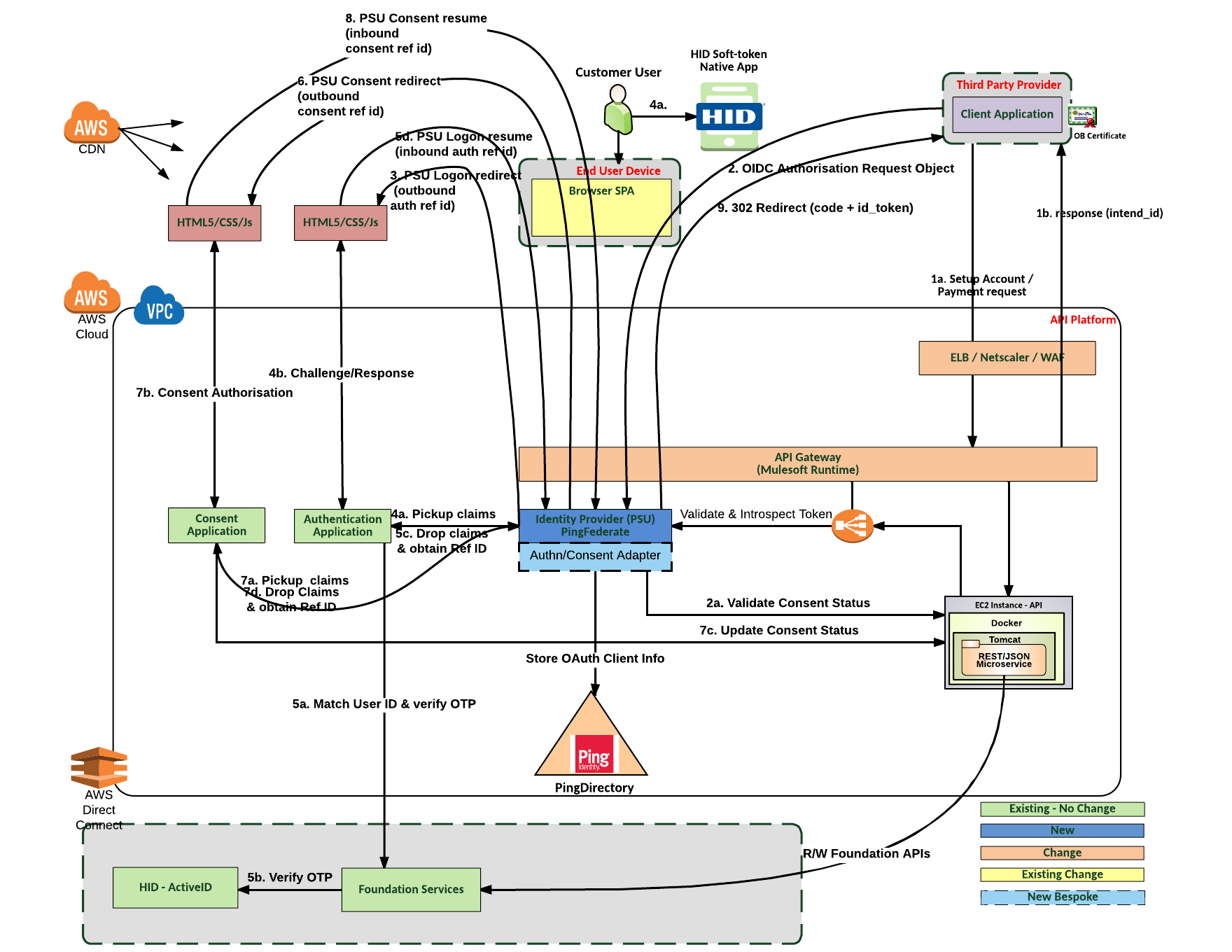
# Integration design – Authentication & consent

The following narrative summarises the steps involved in the authentication and consent customer journey as part of an authorisation grant (not some steps have been simplified to focus more on the authentication and consent authorisation integration):

1. TPP private component initiating an accounts or payments request to setup the transaction following a client credentials grant and receiving an intent\_id
2. TPP preparing and sending OIDC connect request object to the OP authorisation endpoint to initiate the authorisation grant via customer user agent,
3. OP validation of the consent object to check intent – type and status.
4. OP initiates the authentication journey through the reference ID adapter redirecting to the authentication login URL (passing dynamic resume path URL) – this is triggered by a scope based selector as outlined in section 3.4
5. Customer interaction with authentication web application to authenticate themselves (user ID, Password and HID Push Notification Accept through a Signature).
6. Authentication application invokes the PingFederate dropoff endpoint using MA-TLS following successful logon to dropoff user identity attributes.
7. PingFederate returning a short-lived (ephemeral) reference ID value to the Authentication Application.
8. Authentication application builds a redirect URI based upon the OP base URL + dynamic resume path + ref ID.
9. PingFederate updates session status with the authenticated user principal details.
10. OP initiates the consent journey through the reference ID adapter redirecting to the consent URL (passing dynamic resume path URL & Ref ID)
11. Customer user agent redirected to the consent web application to view consent for authorisation.
12. Consent application invokes the PingFederate pickup endpoint using MA-TLS using Ref ID to pick-up user identity and intent id attributes.
13. PingFederate returns the attribute values to the consent application.
14. Customer interaction with consent web application to view and accept consent choosing the correct account
15. Consent application invokes the PingFederate dropoff endpoint using MA-TLS following successful logon to dropoff consent status attributes.
16. PingFederate returns a short-lived (ephemeral) reference ID value to the Consent Application.
17. Authentication application builds a redirect URI based upon the OP URL + dynamic resume path + ref ID.
18. PingFederate updates session status with the consent attributes.
19. PingFederate issues the authorisation code and id\_token to the TPP private component (server)
20. TPP validates the authorisation code and id\_token before calling OP to exchange auth code for an access token.
21. OP validates authorisation code and returns the access token
22. TPP invokes the accounts or payments API to submit the request
23. Mulesoft invokes the PingFederate introspection endpoint to validate the access token and obtain claims
24. PingFederate returns the access token status and a set of related claims to mulesoft
25. Mulesoft enforces the access policies based on the API request and permits the request to pass to the microservice resource server
26. API Resource server processes the API request and returns the API response and payload.

## Authentication & Consent Component Interaction Diagram

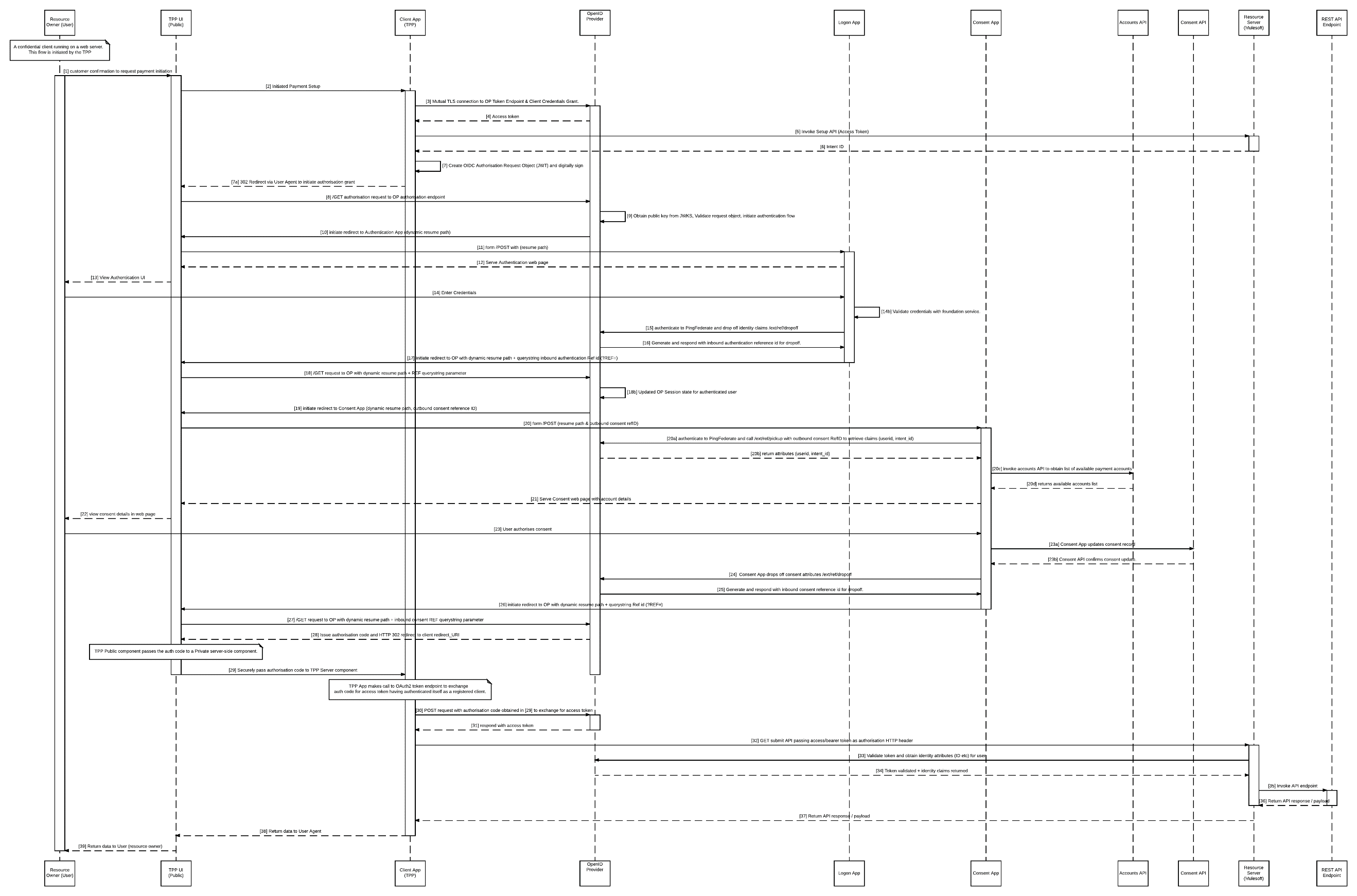
The diagram below depicts a high-level view of the interactions involved during the authorisation grant from an authentication and consent perspective



**Figure 3 – Authentication & Consent Integration Design**

The following UML Sequence Diagram covering the logon and consent journey happy case flow.

## Consent & Authentication Sequence Flow Diagram



## Pingfederate agentless integration kit

##### The PingFederate Agentless Integration Kit includes the ReferenceID Adapter, which enables applications to integrate with PingFederate server acting as either 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 PingFederate will delegate fully the user authentication journey to it.

##### The benefit of this adapter is that it’s a standard off-the-shelf adapter supplied by PingFederate and it follows a lightweight integration approach whereby the Identity Provider application does not need to deploy any agent software in order to interface with PingFederate instead user attributes are passed via direct HTTP calls between the application and PingFederate through a ‘back-channel’ and internal server based interface.

##### For IdP integration, after successful user authentication, the application makes a direct HTTP call to PingFederate to drop-off user attributes about the authenticated user principal and any supporting metadata including authentication time and any pertinent role or attribute based information, which PingFederate temporarily stores, returning a reference (ID) to them in the HTTP response. The IdP application redirects the browser to PingFederate, including the reference to achieve single-sign-on.

##### The technical steps are:

##### Authorise request initiated by TPP client to PingFederate invokes the ReferenceID adapter which issues a 302 redirect to the configured logon application front-end URL passing the reference ID as a query string parameter in addition to dynamic resume path that the authentication application must remember for the purpose of returning once authentication has completed to continue processing – both positive and negative outcomes.

1. The IdP application makes a direct HTTP call to PingFederate /pickup endpoint to retrieve any supporting attributes (as JSON-encoded objects) from PingFederate passing in the outbound reference ID obtained in step 1 above.

**NOTE**The IdP applications must authenticate to PingFederate using one of three mechanisms. If authentication fails, the HTTP request results in an HTTP response 401 – Unauthorized status code message. The authentication method chosen is to use basic authentication (username/password) secured by TLS.

##### PingFederate returns the available authorisation or single-sign-on request attributes and existing session context to the authentication application.

##### The authentication (IdP) application authenticates the user (if not already authenticated).

##### The IdP application makes a direct HTTP call to PingFederate /dropoff endpoint with the user attributes obtained during authentication (as JSON-encoded objects) to PingFederate.

##### PingFederate stores the attributes passed in by the authentication application and responds with a JSON-encoded reference ID string value.

##### PingFederate stores the attributes and returns a reference in the HTTP response to the IdP application.

##### The IdP application redirects the browser to the PingFederate resume path (received in Step **1**) with the reference in the query string. For example:https://pingfederate.example.com:9031/[resume-path]?REF=>

### ReferenceID 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. To use this authentication for PingFederate 6.x and higher, the secondary SSL port must be configured, and application calls must use this port. This is not applicable given the chosen authentication method is to use basic authentication (username/password) secured by TLS which was decided by Capgemini for consistency with the API Platform blueprint.

Your server may already be configured to use the secondary port for other back-channel SSO scenarios (for example, using SOAP). If not, follow this procedure:

1. In the <pf-install>/pingfederate/bin directory, open the file run.properties and change the pf.secondary.https.port value from -1 to a valid port number.

For more information about this property and related configuration settings, see **[PingFederate properties](https://documentation.pingidentity.com/pingfederate/pf81/adminGuide/concept/changingConfigurationParameters.html" \o ")** in the PingFederate *Administrator’s Manual*.

Configuring Certificate Authentication between the client application and the ReferenceID Adapter requires the following steps:



Store the Client SSL private certificate file (sampleClientSSLCert.p12) in the file system and specify the configuration.jsp file CLIENT\_KEY\_FILE\_PATH constant. If the public certificate is not self-signed, additional public certificates may be needed to complete the trust chain.

Import the public certificate into the PingFederate Trusted CAs list using the administrative console, along with any supporting certificates.



Store the PingFederate server public SSL certificate X509 (pfserverSSLCert.crt) file in the file system and point the configuration.jsp file constant SERVER\_CERTIFICATE\_PATH to the SSL certificate X509 file.



Enable PingFederate to use a secondary SSL port. In the run.properties file, set property pf.secondary.https.port to the appropriate port for example, 9032. The default value is -1.

Set the Samples configuration.jsp to the same port value.



Modify the Samples configuration.jsp constants PF\_SECONDARY\_SSL\_PORT and CLIENT\_KEY\_FILE\_PATH.

Set CLIENT\_KEY\_FILE\_PASSWORD and SERVER\_CERTIFICATE\_PATH appropriately and set CERTIFICATE\_AUTHENTICATION to True.

Set the constant SKIP\_HOSTNAME\_VERIFICATION to True if the URL's hostname and the server's identification hostname mismatch, and you want to accept all hostnames.



Configure the ReferenceID Adapter to require a certificate by specifying the allowed subject and/or issuer DN using the administrative console.

Pickup

1. The target application makes an authenticated direct HTTP(S) call to PingFederate to retrieve the user attributes. For example: https://pingfederate.example.com:9031/ext/ref/pickup?REF=ABC123

**NOTE**The applications must authenticate to PingFederate using one of three mechanisms. If authentication fails, the HTTP request results in an HTTP response 401 – Unauthorized status code message. See **[Authenticating to PingFederate](https://docs.pingidentity.com/bundle/agentlessIK13_sm_AgentlessIntegrationKit/page/agentlessIK_c_AuthenticatingToPingFederate.html" \o ")**.

1. PingFederate looks up the attributes (in the above example, referenced by ABC123) and provides them to the target application in the HTTP response. See **[Reference value](https://docs.pingidentity.com/bundle/agentlessIK13_sm_AgentlessIntegrationKit/page/agentlessIK_c_ReferenceValue.html" \o ")** .
2. The target application uses the attributes to create a user session, enabling access to the target resource.

## PingFederate Selectors & Adapters – Custom & Out-of-Box.

This section details the custom and off-the-shelf selectors and adapters used by PingFederate to achieve compliance with the security profile and the functional authentication and consent customer journey.

### Selectors

Authentication selectors provide a plug-in capability for PingFederate to evaluate various conditions related to incoming Oauth2/OIDC and SSO requests. PingFederate comes bundled with a set of authentication selectors including for example, you can create an HTTP header authentication selector to detect mobile browsers, a CIDR authentication selector to evaluate whether the users' IP addresses fall within your internal network ranges, or an HTTP request parameter authentication selector to identify IdP connections based on the PartnerIdpId parameter values provided in the SP-initiated SSO requests. The following selector type has been used in this project:

| **No** | **Selectors Name** | **Custom/OOB** | **Function Description** | **Complexity** | **Implementation Approach** |
| --- | --- | --- | --- | --- | --- |
| 1 | OAuth Scope Authentication Selector | OOB | This selector compares the scope configured with the scope supplied in the OAuth+OpenID connect flow HTTP request. Various instances of this selector can be created to validate different values of the scope. When configured on policy tree can be used to control the path the request can take (like AISP, PISP, failure,  etc..). | Low | This selector is OOB . The selector is just configured on the PF Authentication Policy Selectors configuration page.  The following selector instances are configured for Open Banking project  openid\_accounts\_payments - This selector is used if the scopes supplied are "openid accounts payments"  to fail the request as the OAuth request cannot be initiated for both AISP and PSIP flow  openid\_accounts - This selector is used if the scopes supplied are "openid accounts" to direct the request to take the path configured for AISP  openid\_payments -This selector is used if the scopes supplied are "openid payments" to direct the request to take the path configured for PISP |

### Adapters

Identity Provider (IdP) adapters are a standard way in PingFederate to achieve authentication / authorisation in some way. There are several available IdP adapters supplied in the base product in addition to creating custom adapters using the Ping Identity supplied Java SDK. The following adapter types have been used in this project:

| **No** | **Adapters Name** | **Custom/OOB** | **Functional Description** | **Complexity** | **Implementation Approach** |
| --- | --- | --- | --- | --- | --- |
| 1 | ReferenceID Adapter | OOB - PF supplied integration kit | The Ping Federate Agentless Integration Kit includes the ReferenceID Adapter, which allows developers to integrate other applications with a PF server.  The Authentication and Consent applications are integrated with PF using this adapter.  The **PF redirects**the user to the other application (Authentication/Consent) whose end point is configured in the adapter.  The other application (Authentication/Consent) authenticate itself with PF, and **pickup** the attributes that it needs from PF  Once the other application completes it's process (authenticate the user in case of Authentication application and get user consent in case of Consent application) **drops off** the attributes (like subject, consent details, etc...) to PF. PF stores those attributes and gives back a Reference ID in the response to the other application  The other application redirects the user to PF to continue with the OAuth+OpenID connect flow. | Low | 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  authrefid1 - To integrate with Authentication (Signin) application  consentaisp - To integrate with AISP Consent application  consentpisp - To integrate with PISP Consent application |
| 2 | OB PF Validator Adapter | Custom | This adapter is used to perform pre-authentication validation check on Intent id supplied and tpp.  If the supplied intent id is not a valid one, then this adapter redirects the resource owner to the TPP with error message "Invalid Request"  If the client\_id (TPP) provided has been blocked, then this adapter redirects the resource owner to the TPP with error message "unauthorized client" | Medium | This Adapter is an implementation of IdpAuthenticationAdapterV2 interface of PF SDK. The Adapter instances can be created on Manage Idp Adapter Instances configuration page.  The following instances are created for the Open Banking project  IntendValidator - To validate if the intent supplied in the OIDC Request object is valid. The adapter leverages the /validateIntentId API to perform this check  tppcheck - To validate if the client\_id (TPP) supplied in the OAuth + OpenID connect request is active and not blocked. This adapter leverages the /validateTPP API  Please refer the confluence page <https://raidiam.atlassian.net/wiki/spaces/BOBD/pages/34111493/Authentication+Application+Pre-launch+Validations+of+Intent+TPP+Block+Check> for more information on the /validateIntentId and /validateTPP APIs. |
| 3 | OB PF Fail Adapter | Custom | This adapter is used to supply custom error messages in case of the scope adapter is failed. | Low | This Adapter is an implementation of IdpAuthenticationAdapterV2 interface of PF SDK. The Adapter instances can be created on Manage Idp Adapter Instances configuration page.  The following instance are created for the Open Banking project  invalidscopefail - If the scope supplied in the OAuth+OpenID connect request is "openid accounts payments", which is a failure scenario, then PF forwards the request to this adapter which fails the authentication flow and redirects to TPP with custom error message, that is configured in the adapter configuration page |
| 4 | OB PF Request Object Param Validation Adapter | Custom | This adapter can be used to validate the presence of a parameter in the OIDC request object, and based on configuration in the HTTP request query string parameters.  The name of the parameter to validate, whether to check in HTTP request query string are configurable | Medium | This Adapter is an implementation of IdpAuthenticationAdapterV2 interface of PF SDK. The Adapter instances can be created on Manage Idp Adapter Instances configuration page.  The following instance are created for the Open Banking project  redirect\_uri\_check - This instance of the adapter validates if the redirect\_uri is supplied in OIDC request object (and HTTP request query string parameter - based on the configuration) |

# System Design

## URL Naming conventions

PingFederate runtime engine endpoint URL’s are used to consume the services offered by PingFederate.

There will be a publicly accessible set of endpoints exposed to the internet via a load balancer at different context roots of the same fully qualified domain name.

In addition, 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.

For production there will be a BOI branded entry point for PingFederate runtime engines exposed to the internet as follows:

**auth.boi.ie/authorize**

**secure.boi.com/token**

**Note: both URLs above are pending TDA ratification.**

The rationale for this is:

1. Simple, short and meaningful hostname as it will be highly visible to customer (external) users across all the Bank’s web and API channels in due course – when presented in the browser address bar for example.
2. Identity and security is likely to be the paramount concern of a customer at this stage of the customer journey – therefore it is important to assure customers when presenting their identity credentials during logon activities.

## Design Decisions

##### 

# Component Description

## State management design

The underlying vendor technology supports several state management strategies:

* Cookies on host
  + The PF cookie is generated every time you interact with PingFederate and in general is there to keep track of an SSO event and identify a session.
  + The PF cookie is by default set without domain information in the HTTP header; for example:

Set-Cookie: PF=zOv4xxmzDI2rx1TFBFy78X;Path=/;Secure;HttpOnly

Note the Secure and HttpOnly parameters for maximum security.

* In memory across Worker nodes (PingFederate)

This solution will only maintain state within user cookies and in memory in the individual nodes in the clusters. The other potential state management options will not be supported to ensure delivery has minimal dependencies on other technologies.

PingFederate uses a HyperSQL relational Java database that runs on PingFederate runtime engine nodes and is managed as a 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.

## PingFederate clustering

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. Adding a runtime node is best achieved by snapshot of existing node and copy to new
5. 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 initialisation only).
6. Normal operation is that all runtime nodes interface with each other node.
7. 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).
8. PingFederate makes use of Jgroups for clustering - supports dynamic clustering.
9. Restarting of PF runtime nodes rarely required.
10. How does the Admin know if the config has been pushed and successfully applied - it doesn't.
11. 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.
12. PF cluster state is maintained by the runtimes engines but the extent of the maintenance can be a burden on the cluster itself as a runtime engine – having 4-8 runtimes is deemed OK to operate without a separate state management layer. Need to more carefully consider beyond that.
13. State servers are runtime nodes not visible to the load balancer - dictated by the preferred node indices in the PingFederate run.properties file..
14. State management - each node send and receives to each other node in the cluster.
15. PingFederate Admin node must be a single node for a given cluster - cannot have more than one currently. If relying on the admin API this is a single point of failure.
16. 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 deployment topology and decisions.

Here is a summary of points that capture the deployment approach for Ping components:

1. All PingFederate runtime and admin console nodes will be hosted on Amazon Linux 64bit Virtual Machines (VMs) sourced as an Amazon Machine Image (AMI) running across AWS EC2 instances.
2. PingFederate will be deployed into the application layer and will also run active/active across two or three eu-west region availability zones – decision to be confirmed by BOI and design to be updated to reflect once finalised.
3. The Administrative nodes for PingFederate will be deployed into the application layer.
4. The service ports for each PingFederate runtime and admin instance may be varied to be non-default across each pre-production and production environment should for consistency – in particular the runtime engine ports.

Similarly, for PingFederate

1. Each runtime virtual server has a single active instance. Should that EC2 instance fail for any reason, a new runtime instance shall be automatically created using Capgemini automation toolset.
2. There will be a single VMs for PingFederate admin. Only a single admin console instance can be active in a cluster at any given time.

# system configuration

## Mulesoft Token Provider

##### Runtime level configuration for PingFederate (Front Channel)

##### The host, port for the front channel shall be <hostname>.boi.ie on TCP port 443, secured using a Capgemini certificate authority in the ‘trusted certificate group’ as setup under security settings.

##### Runtime level configuration for PingFederate (Back Channel)

##### This will point to the internal facing load balancer for the primary runtime TLS port of 9031. Note both the load balancer and the PingFederate runtime listener ports are the same. The back channel secure flag will be enabled.

## PingFederate DataSTORES

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

## Listeners & Security Settings (Certificates & Key Pairs)

##### There will be a different key pair created and corresponding digital certificate issued (Capgemini internal enterprise PKI) for the PingFederate Admin Console and Runtime Server. The issuing certificate chain will be need to be added to a trusted certificate group.

##### Note: certificate revocation checking using CRL distribution points or OCSP will be disabled.

## Access token management

##### PingFederate will use an internally managed reference token with a configurable token lifetime and token length; initially 28 char and 120mins respectively.

## Oauth Client (various)

##### A number of OAuth clients will be required for Mulesoft runtimes and OAuth access token introspection calls to PingFederate runtimes.

##### IN each case:

##### It will have a unique client id and secret with only the allowed grant types enabled for that client (based on a least privilege principle).

# Appendix

## Pingfederate endpoints

Below are the application endpoints exposed by the PingFederate runtime cluster that are called by consuming applications, protected by TLS.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Application Endpoint** | **URL** | **Web or**  **API** | **Comment** | **Security Model** |
| **IDP Application Endpoints** |  |  |  | **/IDP endpoints to be exposed in this phase** |
| **Start SSO** | **/idp/startSSO.ping** | **Web** | this is a web request, POST or GET to start an SSO flow | Exposed by Direct via AWS ALB |
| **Start SLO** | **/idp/startSLO.ping** | **Web** | this is a web request, POST or GET to start an SLO flow | Exposed by Direct via ALB |
| **Write Common Domain Cookie** | **/idp/writecdc.ping** | **Web** | only required if supporting the SAML standard for IdP discovery. | Not Exposed by Mulesoft |
| **HTTP Heartbeat** | **/pf/heartbeat.ping** | **Web** | Used to heartbeat the PingFederate runtime engine. Only used by load balancer should be blocked from outside. | Do not expose externally – denied by Mulesoft |
| **Adapter-to-Adapter Mapping** | **/pf/adapter2adapter.ping** | **Web** | Not using this feature | Do not expose externally – denied by Mulesoft. |
| **Get Federation Metadata** | **/pf/federation\_metadata.ping** | **Web** | This can be used with external parties that support exchange of metadata and now also used in the new certificate update process - block for now and reopen later if going to use it | Do not expose externally – denied by Mulesoft. |
| **IdP Services SAML v2.0 Endpoints Single Logout (SLO) Service** |  | **If using a SOAP binding it is an API call if using a POST binding it is a Web browser.** | All of these are standard SAML endpoints and get used depending on the setup of your connections. In general you want to probably leave these alone because it depends on the connection features. If you don't use SLO at all and have no plans then blocking would be OK. |  |
| **Redirect** | **/idp/SLO.saml2** | **Web** | Per above | Not Exposed by Mulesoft |
| **POST** | **/idp/SLO.saml2** | **Web** | Per above | Not Exposed by Mulesoft |
| **Artifact** | **/idp/SLO.saml2** | **API (SOAP)** | Per above | Not Exposed by Mulesoft |
| **SOAP** | **/idp/SLO.ssaml2** | **API (SOAP)** | Per above | Not Exposed by Mulesoft |
| **Single Sign-on (SSO) Service** |  |  | All of these are standard SAML endpoints and get used depending on the setup of your connections. In general you want to probably leave these alone because it depends on the connection features. If you don't use SLO at all and have no plans then blocking would be OK.    It could become a pain to manage these if you want to block, because very much tied to features enabled in connections. |  |
| **POST** | **/idp/SSO.saml2** | **Web** |  | Exposed by Mulesoft |
| **Redirect** | **/idp/SSO.saml2** | **Web** |  | Exposed by Mulesoft |
| **Artifact** | **/idp/SSO.saml2** | **API** |  | Exposed by Mulesoft |
| **SOAP** | **/idp/SSO.saml2** | **API** |  | Exposed by Mulesoft |
| **Attribute Query Service** | **/idp/attrsvc.ssaml2** | **Web** | This service is not really used very much and BOI are not likely to use it. | Exposed by Mulesoft |
| **Artifact Resolution Service** | **/idp/ARS.ssaml2** | **API** | This is tied to the artifact SSO and SLO flows. | Exposed by Mulesoft |
| **OAuth 2.0 Authorization Server Endpoints** |  |  |  |  |
| **OAuth Token Endpoint** | **/as/token.oauth2** | **API** | Client makes HTTP POST request with parameters sent in the body as content-type: application/x-www-form-urlencoded. A successful response to this message will result in a 200 OK HTTP response with the access token returned in a JSON structure in the body of the response. | Exposed by Mulesoft |
| **OAuth Authorization Endpoint** | **/as/authorization.oauth2** | **Web** | For the authorisation code grant type, the client application will redirect the user to the authorization endpoint.  This redirect will contain the applicable attributes URL encoded and included in the query string component of the URL. This will initiate an authentication process using the browser (user agent). Once the user  successfully completes the authorization request, they will be redirected with an authorization code to the redirect\_uri value defined in the authorization request.This can be API or web depending on grant type. | Exposed by Mulesoft |
| **OAuth Grants Endpoint** | **/as/oauth\_access\_grants.ping** | **Web** | This is a web endpoint, page presented to user after authentication of what persistent grants they have authorized so they can revoke. | Exposed by Mulesoft |
| **OAuth introspection endpoint** | **/as/introspect.oauth2** | **API** | This is a web endpoint that accepts form data in an HTTP POST (following client authentication) used by a resource server (RS) client to validate an access token or a refresh token prior to granting access to a protected-resources call. It returns a JSON response. | Exposed by Mulesoft |
| **OAuth token revocation endpoint** | **as/revoke\_token.oauth2** | **Web** | Per RFC 7009, it allows clients to notify the authorisation server that a previously obtained refresh or access token is no longer needed – the request (HTTP POST form data only) invalidates the actual token and possibly other tokens based on the same authorisation grant. | Exposed by Mulesoft |
| **OAuth Access Grant Management Service** | **/pf-ws/rest/oauth/clients/{clientId}/grants/[/{grantId}] or**  **/pf-ws/rest/oauth/users/{userKey}/grants/[/{grantId}]** | **API** | Enables the retrieval and revocation of access grants for a particular client or user | Do not expose externally – denied by Mulesoft. |
| **OpenID Provider Endpoints** | **A public endpoint that provides configuration information for the oauth clients to interfaces with PingFederate using the OpenID Connect protocol** |  |  |  |
| **OpenID Well known configuration endpoint** | **/.well-know/openid-configuration** | **API** | Returns well known configuration of OpenID provider – JSON document | Do not expose externally – denied by Mulesoft |
| **JSON Web Key Set** | **/pf/JWKS** | **API** | Returns a set of public keys in a JSON document | Do not expose externally – denied by Mulesoft |
| **PingFederate Revoked Sessions endpoint** | **pf-ws/rest/sessionMgmt/revokedSris** | **API** | Takes POST or GET with JSON formatted name value pair:  Post adds a session identifier of abc123 to the revocation list : {“id”:”abc123”}. A GET request with the same session identifier is done by adding the abc123 onto the URI at the end and if found on the revocation list, endpoint returns a 200 – OK response. | Do not expose externally – denied by Mulesoft |
| **OpenID UserInfo endpoint** | **idp/userinfo.openid** | **API** | Clients send a UserInfo request using either HTTP GET or POST with access token as bearer token in ‘authorisation’ header field to PingFederate in exchange for a list of claims about a user in JSON response. | Exposed by Mulesoft |

 In summary, the endpoints tabularised above will be dealt with as follows:

1.    API only endpoints that take JSON input payloads will not be exposed externally and denied by the Mulesoft through a white-list function from which they would be excluded.

Should it be necessary to expose API endpoints externally in future phases, they will need to be secured following standard web service security gateway technologies.

1. Web only and API labelled endpoints in the table above will be exposed by Mulesoft and secured through the Citrix Netscaler acting as a Web Application Firewall – WAF.

Note: for the reference id adapter,

The PingFederate URL an application uses for dropping off attributes is:

http[s]://<pf-host>:<pf-port>/ext/ref/dropoff

The PingFederate URL an application uses for picking up attributes is:

http[s]://<pf-host>:<pf-port>/ext/ref/pickup

In addition, the authentication and consent application redirects the browser to the PingFederate resume path with the reference in the query string. For example: <https://pingfederate.example.com:9031/[resume-path]?REF>=>

Note this resumepath endpoint is exposed directly and not proxied by Mulesoft.

# Definitions, Acronyms and Abbreviations

This section defines all terms, acronyms and abbreviations used in this document.

|  |  |  |
| --- | --- | --- |
| No. | Term / Abbreviation | Description |
| 1 | OAuth2 | Open Standard Authorisation Framework (RFC 6749) |
| 2 | Bearer Tokens | Per RFC 6750, OAuth enables clients to access protected resources by obtaining an access token which is a string representing an access authorization issued to the client. A bearer token is a security token with the property that any party in possession of the token (a "bearer") can use the token in any way that any other party in possession of it can. |
| 3 | URI | Uniform Resource Identifier - is a string of characters used to identify a name of a resource. |
| 4 | URL | Uniform Resource Location - is a fully qualified reference (an address) to a resource on the Internet. It comprises a protocol identifier and a resource name. |
| 5 | IAM | Identity & Access Management |
| 6 | AWS | Amazon Web Services |
| 7 | AZ | Availability Zone – A standalone data centre forming part of an AWS region. |
| 8 | EC2 | Elastic Cloud Compute from AWS. |
| 9 |  |  |
| 10 |  |  |

## PSD2 Glossary

##### 

## References

This section lists all the applicable and reference documents, identified by title, source and date.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ref | Title | Source | Date | Issue |
| RFC 6749 | OAuth2 Standard | IETF | N/A | N/A |
| RFC 6750 | OAuth: Bearer Token Usage | IETF | N/A | N/A |
| RFC 7662 | OAuth 2.0 Token Introspection | IETF |  |  |
| OB Security Directory | Open Banking Directory | Open Banking |  | 1.1.0 |
| OB Security Profile | Open Banking Security Profile | Open Banking | 29/09/17 | 1.1.0 |
| OB Payment Initiation API | Open Banking Payment Initiation API | Open Banking | 31/08/17 | 1.1.0 |
| OB Account & Transaction API | Open Banking Account & Transaction API | Open Banking | 31/08/17 | 1.1.0 |
| PSD2 RTS | PSD2 RTS - Final Report - Draft Regulatory Technical Standards on Strong Customer Authentication and common and secure communication under Article 98 of Directive 2015/2366 (PSD2) | EBA | EBA/RTS/2017/02 | Final  Draft |

# Appendix – sequence diagrams

## TPP client registration

##### 

## Payment initiation – setup

## authorise consent

##### 

## Payment submission

##### 