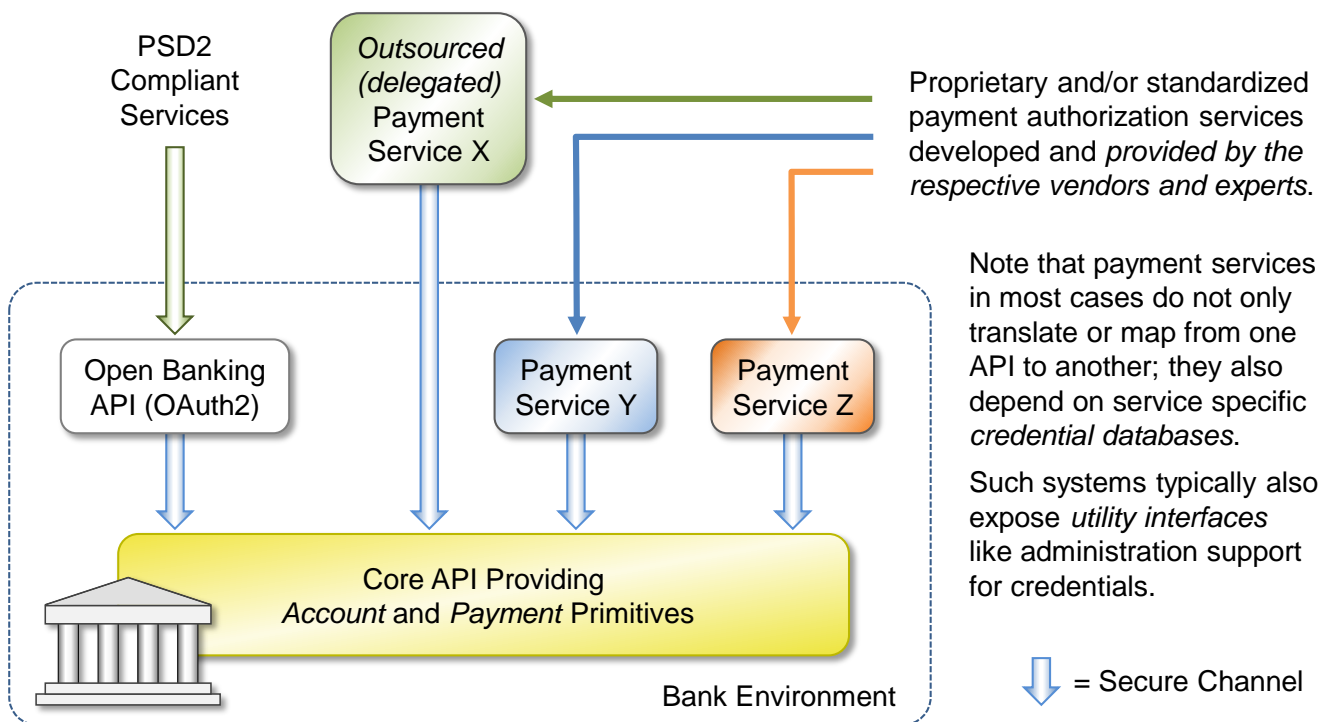


# Revised Open Banking Architecture

Background: As you may have noted the combination of the evermore popular digital wallets and Open Banking do not seem to happen. The reason for that is quite simple: the integrated SCA concept in current Open Banking APIs interferes with wallets like Apple Pay that have their own SCA solution.

To cope with this problem, specific wallet APIs have been proposed. Although working, such APIs would effectively block third party developments (like the EU Wallet), as well as greatly complicate implementations, testing, and rollout.

This document outlines another solution which could be illustrated like this:



This solution builds on a *layered model* for the Open Banking API, where a fixed Core API, supports multiple, trusted application specific services. Since the Core API is a privileged service, the Core API is never exposed directly to external services (the trusted application services function as mediators). To maintain system integrity, the trusted application services and the Core API, communicate through a *mutually authenticated* TLS channel. Judicious use of *logging* is highly recommended. Note that the Core API does not authenticate users (this is supposed to be performed by the trusted application services).

The Core API is exposed as JSON based REST services, enabling loosely coupled application services.

Summary:

- The non-monolithic nature of this arrangement simplifies Open Banking API documentation, development, and testing.
- Loosely coupled application services can be built on vendor specific platforms. The loose coupling also enables application services to be shipped as boxed systems, docker images, or virtual machines.
- The described architecture permits support for Core API sandboxes, making payment service development in reach for any entity. Using sandboxes, payment service iterations become simple and fast.

Write Once – Deploy in Any Compliant Bank