OpenLegacy
Digital to the Core

Case Study

Top 10 Global Bank Implements Six Key Global APIs and Microservices in Just Two Weeks

Bank achieves goal of “Global APIs,” plus improves API performance 5-7x

Financial | IBM System i | Mainframe | Core Banking Systems

Founded over 150 years ago, this multinational bank is one of the world’s largest banking and financial services organizations, serving tens of millions of customers worldwide. With thousands of offices in over 70 countries, in both established and emerging markets, the bank aims “to be where the growth is, connecting customers to opportunities.” Building a strong central entity with a global presence that suits every country it serves is front and center to the bank. This means rapidly developing new consumer-facing applications and business functionality, and responding to changing market and competitive dynamics in an agile fashion.

The Challenge

From the outside, the bank looked like a global bank—but from an IT point of view, it operated in technical, geographical, and business function silos. Like many banks, almost each country had its own technology stack and back-end setup. The backbone is a core banking system (CBS) deployed in 70 countries. However, each country runs a customized version of the CBS tailored to its needs, business processes, and local regulations. Beyond the CBS, the environment includes both IBM i (AS/400), mainframe and other home grown complex backend applications.

Demand for digital, global services led to a backlog of 100+ foundational APIs necessary to build new applications and customer experiences. Nearly 200 developers were working on the project for over a year, using a popular product for API gateway and orchestration. The product did not specialize in legacy (core) applications running on IBM i and mainframe platforms, and came short of addressing the bank’s needs. Specifically, the product could not generate APIs exposing RPG programming language programs, but could only manage and expose existing APIs. Hence, the bank’s developers wrote additional IBM i code in RPG in order to expose functionality. In other cases, they changed existing code, an invasive practice for applications in place for many years. Beyond exposing functionality, there was substantial time required for testing and regional certification which slowed things down even further. Ironically, a tool designed to shorten development time ended up creating additional manual effort.

Another critical requirement was creating a Global API: A unified API with the same end-user experience no matter the country, region, or underlying technical environment. From a business standpoint, this requires the extraction of common logic and functionality that serves as a single launch point for new products and services that are consistently rolled out in a unified, global manner.

To implement this vision, the bank needed a routing mechanism. If an API call came in, the system should route it to the right country. The bank wanted to implement a set of specifications—like “Balance Inquiry,” “Money Transfer,” “Retrieve customer info”—once and have it usable for all countries.

The Solution

After a year of limited results, the bank turned to OpenLegacy. A two-week workshop started with the bank's IT leaders describing their current "cumbersome, complex" architecture. Most notably, the bank wanted to loosely-couple the legacy platform. It was a monolithic approach with several gateways and middleware layers. There was a lot of application logic running on the IBM i, such as user channel logic, teller permissions, and transfer limitations. Beyond the monolithic setup and the limitations it imposed, the middleware and gateway products were also expensive to maintain.

OpenLegacy's API integration team worked out an alternative architecture, showing it was possible to eliminate all the intermediate layers and the channel logic. Instead, the bank could directly expose the IBM i transactions as a Java application co-located with the new applications and logic. Now the bank executes transactions and the generated microservices takes care of the complexity of calling the legacy system for any required information.

The bank's leaders are realizing the short- and long-term benefits of the new Global API architecture to meet their ever growing needs for new products and services in the countries they serve.

The Result

Automation led to increased development speed

As a leading analyst commented when vetting OpenLegacy's technology, "your simplicity is brilliant." OpenLegacy generates standard, lightweight code for microservices, without modifying the backend IBM i and mainframe applications. OpenLegacy's automatic generation of the microservices shortened the development cycle from weeks to days per API.

Eliminated backlog with agile development and standards

Development time was much shorter, thanks to OpenLegacy's automation, standardization, and agile methodologies. API standardization is important for this bank, and OpenLegacy's automated process quickly generated code based on user-customizable standards. Automatic generation of standard code is better than hundreds of developers each using personal coding style and preferences. Once the organization defined its coding preferences, all generated APIs and services complied with these definitions.

Simplicity led to 5-7X performance improvements

OpenLegacy's architecture does not involve middleware layers such as ESBs and MQ. Its architecture calls directly from the API to the legacy system thus leading to impressive performance improvements. Transaction response time improved by 5-7X from 350ms to 50-70ms.

About OpenLegacy

OpenLegacy's Digital-Driven Integration enables organizations with legacy systems to release new digital services faster and easier than ever before. Connecting directly to even the most complex core systems, OpenLegacy automatically generates the digital-ready components needed to integrate legacy assets into exciting new innovations. With OpenLegacy, industry-leading companies release new apps, features, and updates while spending a fraction of the time and resources, so they quickly and easily become digital to the core.