From Spreadsheets to Service Domains

How Backbase industrialized core-banking integrations with BIAN – and mapped the standard's own gaps along the way



The Challenge

Backbase builds the platform that Tier 2 to Tier 4 banks use to modernize. Its Ecosystems division produces integrations into the software those banks already run: core banking systems, CRMs, fraud detection, payment processors. The faster those connections are built, the faster a bank can move.

The bottleneck was never the engineering. It was a spreadsheet. Every integration began with business analysts copying API specifications out of a YAML file and into a spreadsheet by hand, then mapping each attribute across to the vendor's API. Error rates were high, mappings varied with whichever subject-matter expert happened to be in the room, and developers translated spreadsheet rows back into code by hand.

Up to three weeks went into the mapping of a single integration. For a full core banking integration, total elapsed time regularly exceeded six months – most of it spent not on building anything, but on agreeing what mapped to what.

The Solution

Backbase treated the mapping problem as a product in its own right. Connector Studio ingests API specifications automatically, proposes the mappings, and generates the integration code from them.

The BIAN Service Domains and Business Object Model supply the fixed point. Grand Central, Backbase's integration platform, exposes Unified APIs built on BIAN, so customer data in a core, a CRM or a payments application is reached through one contract rather than a bespoke one per component.

© 2026 BIAN e.V.

Inside Connector Studio

A parser reads YAML specifications straight into a structured database, so analysts work with live attributes rather than maintaining copies. Models trained on previous mappings propose how each attribute corresponds to the vendor's, and analysts review and approve instead of transcribing. Every approved rule is stored centrally, so the knowledge base grows with each integration and dependence on individual specialists falls.

From there the mappings become code directly, using Jolt transformations and Apache Camel, with automated validation before handover. The step that used to consume three weeks now feeds the next integration rather than starting from a blank sheet.

Why a Standard, Not a Home-Grown Spec

Most integration platforms carry a unified API standard of their own – Mulesoft and PortX among them. Backbase weighed building one the same way, by comparing the major core vendors' APIs and aggregating them into a house specification. It chose BIAN instead for three practical reasons: customers were already asking about it, colleagues had worked with it before, and it covered a larger share of the domains needed for the first wave of integrations. The migration ran from proprietary legacy APIs to unified BIAN Service Domain APIs, with the BIAN team consulted on certification of the product.

Where Backbase Had to Go Beyond the Standard

Adopting BIAN was not friction-free. Backbase is unusually direct about the three areas where it had to extend the framework to make it work in practice.

Banking Capabilities and Beyond

BIAN's service domains focus on core banking-system capabilities. Some user-side journeys, such as adding cards to digital wallets, required Backbase to extend beyond the standard.

APIs

Pagination and sorting had to be added to API endpoints where needed.

Business Object Model

Over the last few years, BIAN has made significant progress in mapping service-domain attributes to Business Object Model data types.

While BIAN covers the full banking landscape, the depth of coverage varies by functional area. Most areas now have detailed definitions of data types and enumerations, while others are still being expanded to the same level of detail.

None of this stopped the project. It is recorded here because it reflects the honest shape of adopting a standard that is still growing, and because the gaps identified by members are precisely the gaps the standard can go on to close.

© 2026 BIAN e.V.

Status & Targets

80%

Targeted reduction in manual mapping effort per integration

Live

In production, roll-out starting with the analysts in Ecosystems

3 weeks

Manual mapping per integration before Connector Studio

More connectors

Higher throughput of published ISV integrations, with a sizable effect on revenue

> *While Connector Studio remains Backbase IP, we believe in sharing our knowledge, ideas and learnings with wider industry audiences.*
>
> – Backbase, BIAN Transformation Awards 2025 submission

What's Next

Connector Studio currently serves Backbase's own developers; the stated vision is to open it to anyone building ISV integrations on Grand Central, so the marketplace grows rather than the internal backlog. Once a mapping is complete, the next step is to generate the whole integration service from it, calling BIAN service domains through MCP servers. Two areas of coverage are being extended: more service domains, and Service Domain Events, where support is currently limited.

The Takeaway

Backbase did not adopt BIAN to modernize a bank. It adopted BIAN to stop rebuilding the same translation over and over – and found that a shared vocabulary is worth most precisely where the work is dullest. There is a second-order effect worth noticing: every connector built this way carries BIAN into another vendor's product. The standard proved itself in the spreadsheet, not the architecture diagram.

Every case study in this series began as an award entry. The BIAN Transformation Awards 2026 are open.
Entries close on March 16, 2027 – find the submission area at bian-services.com

This case study is based on Backbase's submission to the BIAN Transformation Awards 2025, in the "Best Use of Innovative Technology" category.

© 2026 BIAN e.V.