International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V72I6P108

Volume 72 Issue 6, 59-63, June 2024
© 2024 Seventh Sense Research Group®

Review Article

Architectural Considerations for Integrating with Finacle

Prasenjit Banerjee[1], Rajarshi Roy[2]

[1]*Technical Architect Director, Salesforce, IL, USA.*
[2]*Discover Financial Services, IL, USA.*

[1]*Corresponding Author : prasenjit.banerjee@salesforce.com*

Received: 23 April 2024 Revised: 27 May 2024 Accepted: 07 June 2024 Published: 15 June 2024

Abstract - *In today's world, every organization uses several different applications across the customer lifecycle journey. Core banking application is like the central nervous system of a bank. It needs to be integrated with all the peripheral systems of the bank. These systems include the CRM system, the internet banking systems, Mobile apps, to back-end systems used by banks for their payment reconciliation or connect to ATM networks. All these systems have different integration requirements. Some of these systems, such as ATMs, need the information in real time or as latest as possible. At the same time, other systems such as printing a checkbook, may wait for the end of the day. In this article, we have explored the various ways to Integrate applications with Finacle. Finacle provides a host of different options for Integrating with it.*

Keywords - *Finacle, Finacle integrator, API, System integration, Middleware, Service oriented architecture.*

1. Introduction

Finacle is a core banking solution developed by Infosys. It is currently licensed and maintained by Edgeverve Systems, a wholly-owned subsidiary of Infosys Technologies. Limited. Finacle started as a small project to serve the core banking needs of Canara Bank.

Later, Infosys saw potential in making it a full-fledged core banking service product in 1999. Finacle is now used as the core banking platform by more than 2000 banks and financial services institutions across 100 countries. It has 1.08B customers and 1.3B bank accounts managed by Finacle.

While Integrating with an application as mammoth as the size and scale of Finacle, Architects at any banking or financial organization need to put together a solid thought process on identifying the needs of these connecting systems. Some due diligence on the following information will be helpful in making a data-oriented decision on Integration patterns that need to be applied.

1.1. Considerations for Deciding the Integration Design

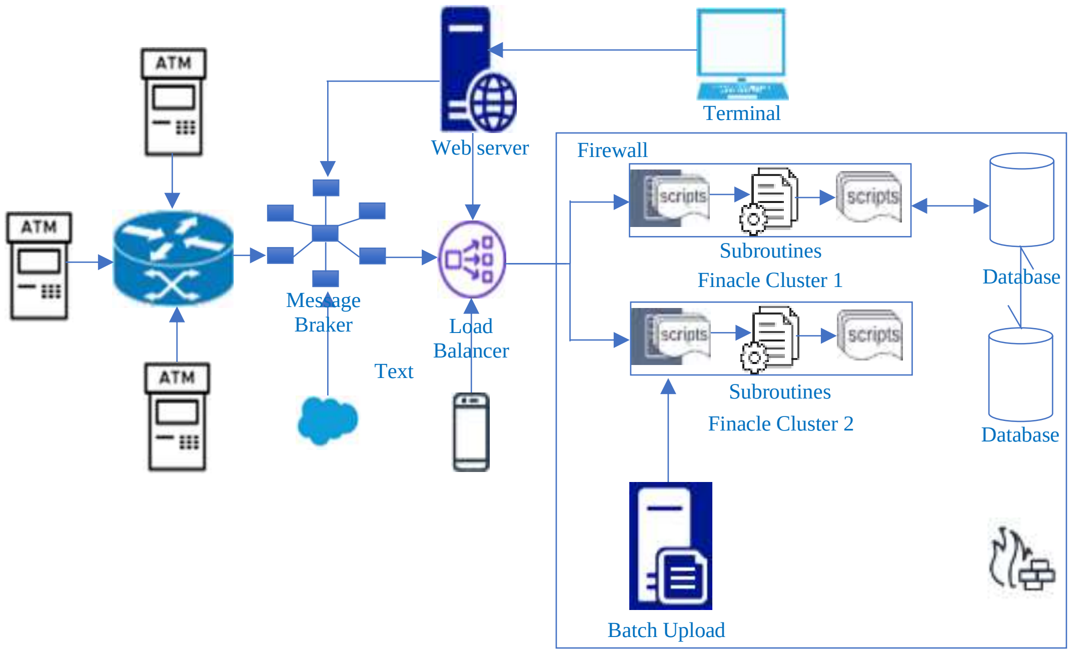
- Transactions per second [1]
- Payload size of the data integrated [2].
- Acceptable latency of the data interchange
- How fresh the data needs to be.
- Security requirements
- Network boundary will it cross [3].
- Bulk or Incremental
- Robust Automated Validation Framework
- Fault Tolerance and reliability

Table 1. Integration patterns for integrating with finacle

Design Pattern	Use case	Consideration
Request-response	Fetching customer balance for CRM application	Transaction per second (TPS), payload size and latency
Batch	Uploading Customer information for statement generation [4]	Bulk or Incremental, Times of Day, error and retry, Data stale
Publish-subscribe	ATM Withdrawal, fraud detection	Reliability, Scalability
Point to point	Loan origination system transferring lead data to Finacle	Latency and resilience

1.2. Finacle Support for Modularity and Extensibility

Although the core engine of Finacle was built more than 25 years ago, the architects at Finacle made some very important architectural decisions, which paid high dividends later. First, they created a Service based architecture, a time when such a term did not even exist. They built extensible subroutines that allowed every event to be customized without tampering with the core functionality. Let us understand what that means. Finacle core functionalities were created in a modular fashion using standard C/C++ routines or functions that catered to a unit level of functionality [5]. The subroutines can be considered as Finacle building blocks. These subroutines are callable, which means they can be invoked from other subroutines and from customized scripts applied on top of product code. These subroutines are extensible. This means that although these routines themselves cannot be modified, several custom events can be added before the after the invocation, which allows for custom business rules to be applied or, in some cases, overrides the default behavior of the product logic. This modularity and composability allowed Finacle to expose these business capabilities as services. These Services were mainly callable over the RPC protocol [6].



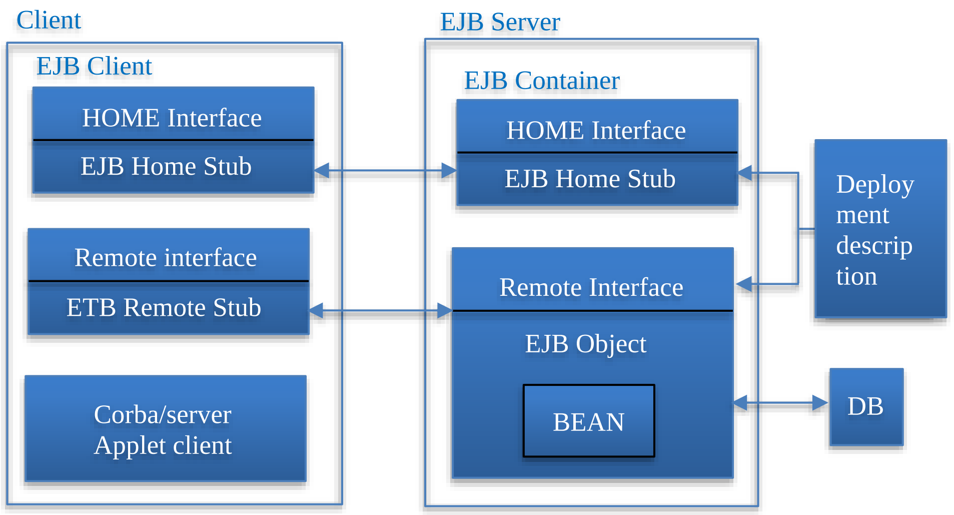


Fig. 2 A working model of EJB-based web service

1.3. Finacle Integrator Framework

Finacle Integrator is a java module that is built on Enterprise Java Beans (EJBs) that allows the extension of shared objects as web services over the TCP/IP network. Finacle Integrator allows Finacle to extend the service functionality beyond the Finacle boundary while observing a common web service protocol. Finacle implements a reliability pattern with Java Messaging Service (JMS).

Finacle implemented a reliability pattern with Java Message Service (JMS) that is essential for ensuring that messages are processed reliably in a distributed system.

JMS provides a common way for Java applications to create, send, receive, and read messages, allowing for loosely coupled, reliable, and asynchronous communication [8].

To ensure reliability in Finacle Integrator, JMS was leveraged with the following key strategies:

- Ensured that messages are set to be persistent. Persistent messages are stored by the JMS provider until they are successfully delivered to the consumer, surviving provider failures.
- Finacle controls how the consumer acknowledges receipt of a message. This can be automatic or managed by the application for more control over the process. Finacle allows auto-acknowledgement; the Finacle acknowledgement EJB automatically acknowledges a consumer's receipt of a message and allows client acknowledgment. The client acknowledges the message by calling the message. Acknowledge () method [9].
- Finacle follows ACID properties of Transactions to ensure that a series of operations are completed successfully before committing them. If a failure occurs, the transaction can be rolled back to avoid partial processing of messages. It rolls back the transaction in case of an error.
- Configure redelivery policies on the JMS provider to handle message redelivery in case of consumer failure. This includes settings for maximum redelivery attempts and delay intervals.
- Use a Dead Letter Queue (DLQ) to handle messages that cannot be delivered or processed after several attempts. This allows for manual intervention or automated processing to handle failed messages.
- Set a time-to-live for messages to ensure they are not delivered if they cannot be processed within a specific timeframe. This is useful for time-sensitive messages [10].
- Implement robust error handling and logging mechanisms in your message consumers. This helps in diagnosing issues and taking corrective actions for message processing failures.
- Design your message processing logic to be idempotent, meaning that receiving and processing the same message multiple times does not result in incorrect behaviour or side effects [11].
- Consider the scalability of your JMS setup. Using Message-Driven Beans (MDBs) in an Enterprise JavaBeans (EJB) container or configuring multiple consumers for a queue can help distribute the load.

Integration via Synchronous calls over HTTP

Finacle Integrator allows for connecting to the Service Objects (SOs) via SOAP. The Finacle Integrator acts as a middle layer between the SOs via an EJB interaction that allows these objects to communicate via SOAP web services, although they are not typically designed to communicate over HTTP vis TCP/IP protocol. Thus, all the integration with internet banking and mobile banking platforms and apps can be done using the Finacle Integrator module, and it scales well.

The synchronous API call over Http/Https to integrate Finacle closely resembles a Service Oriented Architecture (SOA). Finacle publishes a catalogue of such shared objects that can be used as services. These Shared objects are wrapped with standard EJB templates and exposed as SOAP web services.

The contracts for these services are available in the internal registry very similar to a UDDI registry of the organization. Although there was no formal policy-based governance in place, these APIs can be secured through authentication and authorization mechanisms by integrating with an external identity provider. These services are a huge asset for any customer as these services can be leveraged as a hook that is available to systems outside of Finacle.

These services allow for service-to-service orchestration, allowing external systems to perform complex business processes and functions. While these services have a fixed schema in which they expect their inputs, these services can transform data from one format to another although the most common formats are XML and JSON-based message formats.

This is where a smart automated Validation Framework should be used during the integration as this Architecture is very conducive to such a framework to not only get the maximum value out of your Validation process but also save a lot of time in the validation process [12].

1.3. Factors Driving the Architectural Decisions

When designing a data integration system with Finacle, considering factors such as transactions per second, payload size, acceptable latency, data freshness, security requirements, network boundaries, data transfer methods (bulk or incremental), and system fault tolerance and reliability is crucial.

These parameters signixficantly influence the architecture, technology selection, and implementation strategy of the system. Here is an inference that considers these points:

Table 2. Implication and recommendation on the design decisions

Factors	Implications	Recommendations
Transactions Per Second	High TPS requires scalable solutions.	Evaluate downstream system scalability, potential Finacle bottlenecks, and technology stack's throughput capabilities.
Payload Size	Large payloads necessitate robust data platforms to be aware of Finacle limitations.	Assess data storage solutions and network capacity for efficient data handling and transfer.
Acceptable Latency	Low latency demands real-time processing frameworks.	Determine real-time processing needs versus batch processing feasibility and the impact on user experience.
Data Freshness	The need for current data indicates real-time data exchange.	Consider the balance between real-time data processing and system resource utilization.
Security Requirements	High-security calls for encryption and compliance with standards.	Implement robust security measures and ensure compliance with relevant data protection regulations.
Network Boundary	Public network transactions require enhanced security measures.	Analyze network architecture for secure data transmission and identify potential vulnerabilities.
Bulk or Incremental	Bulk transfers fit with batch processing, while incremental updates benefit from CDC mechanisms.	Decide based on data volume, frequency of updates, and system capabilities for handling data ingestion.
Fault Tolerance	High availability and reliability demand robust infrastructure and Finacle is very robust on that subject.	Ensure system redundancy, implement failover mechanisms, and consider the impact of downtime on operations.
Robust Validation	Customizations require Quick and Robust Validation mechanisms for faster go to Market.	Build a Smart Automation Framework during the implementation phase itself and Maintain it for Incremental Customizations in Future.

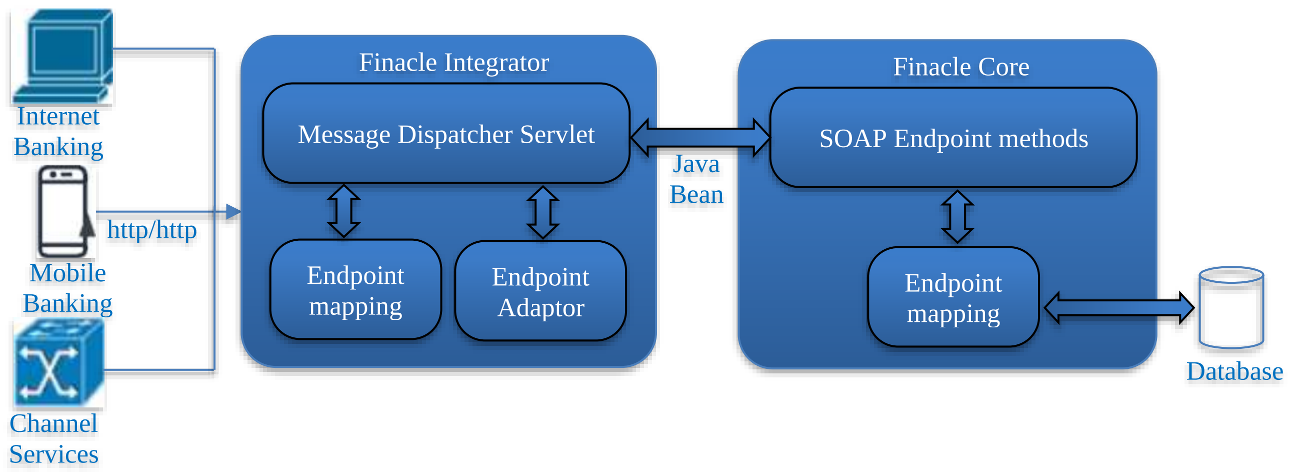


Fig. 3 SOAP web services using finacle integrator

2. Cloud-Native Design Considerations

While banks have the strongest security requirements, nowadays, information security teams of several banks can provide green signals to host their core banking solutions in the cloud. So, there is a strong market demand to provide a core banking solution as a service. The Finacle team can cope with design challenges and has been able to host Finacle in a multi-tenant environment with robust security features. This allows banks to reduce the total cost of ownership while availing the benefits, the cloud provider has to offer. A cloud-native full-hosted solution takes care of the high availability of the solution. It helps to scale the infrastructure on demand, based on consumption. The adoption of a microservices-based approach on top of a Kubernetes-based deployment allows for API-based integration over HTTP/HTTPs. These take care of the governance of the services through an API management layer. Central policy-based governance allows for the application of tight security policies to avoid spike control and denial of service attacks.

3. Conclusion

The integration of Finacle, a leading banking solution, into an organization's existing systems and processes represents a strategic step towards achieving high levels of operational efficiency, enhanced customer experience, and innovative financial service offerings. As Finacle offers a comprehensive suite of banking functionalities, from core banking solutions to digital engagement platforms, its integration is not just a technological upgrade but a transformative initiative that can drive an institution towards global banking standards. Key considerations, such as transactions per second, payload size, acceptable latency, data freshness, security requirements, network boundaries, the choice between bulk or incremental updates and a Robust Smart Automated Validation Framework play a crucial role in the successful implementation of Finacle. These factors must be meticulously planned and managed to ensure that the integration process is smooth, scalable, and secure, thereby enabling seamless operations and real-time data processing capabilities.

Moreover, the fault tolerance and reliability of the system are paramount, necessitating a robust infrastructure that can withstand various challenges and ensure uninterrupted service delivery. By addressing these considerations, institutions can leverage Finacle to not only streamline their operations but also to innovate and create personalized banking experiences for their customers. In conclusion, the integration of Finacle is a complex yet rewarding endeavor that requires careful planning, consideration of several technical and strategic factors, and a commitment to excellence. When done successfully, it positions financial institutions to take advantage of the evolving digital landscape, meet the growing demands of their customers, and stay competitive in the global market.

References

[1] Noussair Fikri et al., "An Adaptive and Real-Time Based Architecture for Financial Data Integration," *Journal of Big Data*, vol. 6, pp. 1-25, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[2] Norbert Bieberstein, *Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap*, FT Press, 2006. [Google Scholar] [Publisher Link]

[3] Latha Srinivasan, and Jem Treadwell, "An Overview of Service-Oriented Architecture, Web Services and Grid Computing," *HP Software Global Business Unit*, vol. 2, pp. 1-13, 2005. [Google Scholar] [Publisher Link]

[4] Tony Chao Shan, and Winnie Wei Hua, "Service-Oriented Solution Framework for Internet Banking," *International Journal of Web Services Research*, vol. 3, no. 1, pp. 29-48, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[5] Peter Gomber et al., "On the Fintech Revolution: Interpreting the Forces of Innovation, Disruption, and Transformation in Financial Services," *Journal of Management Information Systems*, vol. 35, no. 1, pp. 220-265, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[6] Markos Zachariadis, and Pinar Ozcan, "The API Economy and Digital Transformation in Financial Services: The Case of Open Banking," *SSRN*, pp. 1-28, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[7] Ummu Hani' Binti Hair Zaki, "*Web Service Architecture for Scholarly Publication*," PhD Dissertation, Universiti Teknologi Malaysia, 2016. [Google Scholar] [Publisher Link]

[8] Fatna Belqasmi, Roch Glitho, and Chunyan Fu, "Restful Web Services for Service Provisioning in Next-Generation Networks: A Survey," *IEEE Communications Magazine*, vol. 49, no. 12, pp. 66-73, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[9] Felipe Osses, Gastón Márquez, and Hernán Astudillo, "An Exploratory Study of Academic Architectural Tactics and Patterns in Microservices: A Systematic Literature Review," *Advances in Software Engineering at the Ibero-American Level*, pp. 71-84, 2018. [Google Scholar] [Publisher Link]

[10] Konstantinos Vandikas, and Vlasios Tsiatsis, "Microservices in IoT Clouds," *Cloudification of the Internet of Things*, IEEE, pp. 1-6, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[11] Davide Taibi, Valentina Lenarduzzi, and Claus Pahl, "Architectural Patterns for Microservices: A Systematic Mapping Study," *CLOSER 2018: Proceedings of the 8th International Conference on Cloud Computing and Services Science*, Funchal, Madeira, Portugal, 2018. [Google Scholar] [Publisher Link]

[12] Emirates NBD Bank Achieves End-to-end Traceability and Efficiency by Implementing Finacle Core Banking Automation Testing Solution. [Online]. Available: https://www.testhouse.net/wp-content/uploads/2019/10/Finacle-Core-Banking-Automation-Testing-Solution-for-Emirates-NBD-Case-Study-Testhouse.pdf