Infosys cobalt

WHITE PAPER

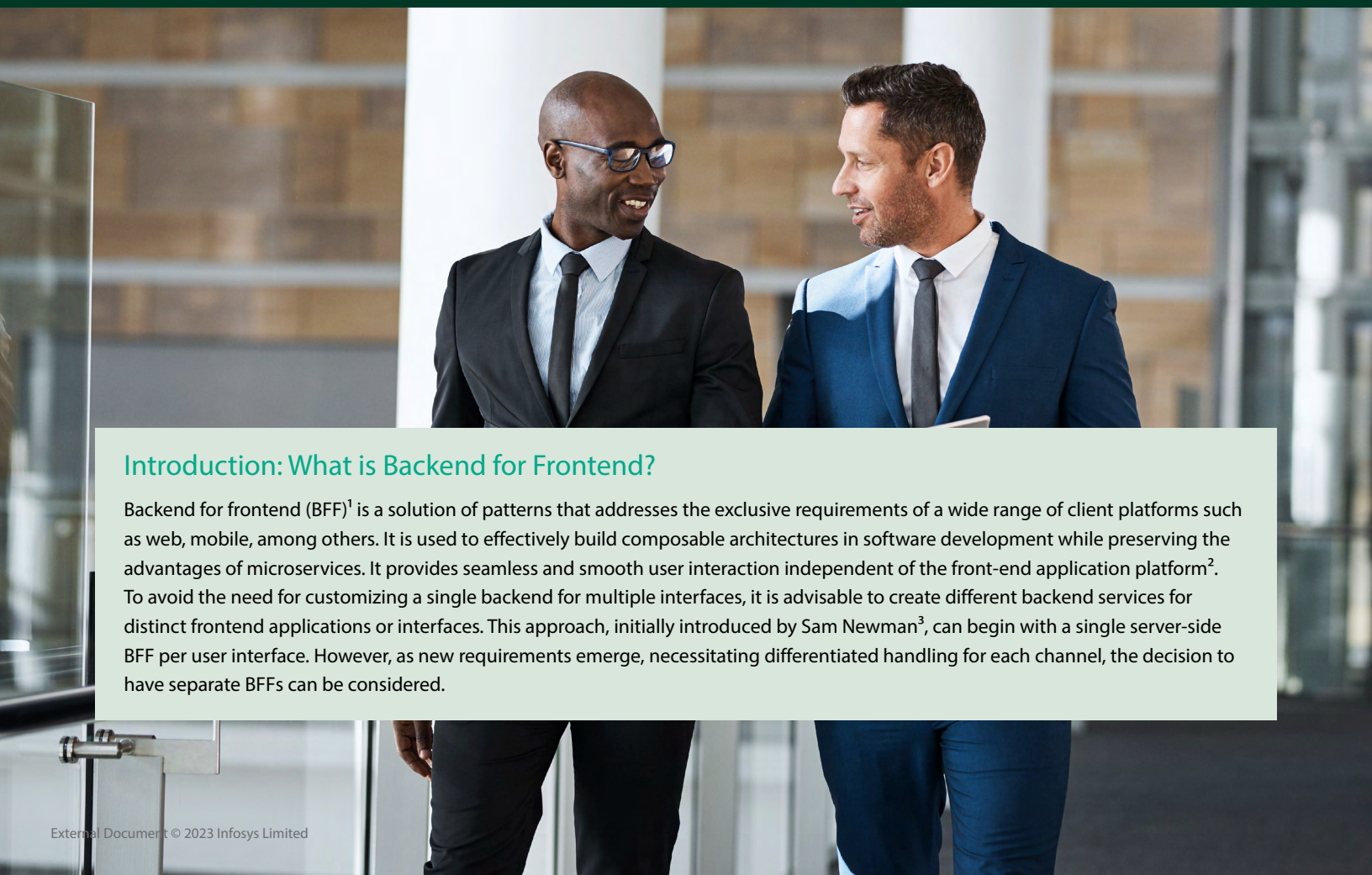
BACKEND FOR FRONTEND: TAILORING BACKEND SERVICES FOR ENHANCED FRONTEND EXPERIENCE IN TELCOS

Abstract

Business strategies today aim to provide a consistent buying experience across all channels. Salesforce Communication Cloud is no exception, serving as a backend application programming interface (API) for various non-native channels. In an omnichannel interaction, having one common backend API adds to the overhead on the backend APIs and the channels. This overhead arises from large payloads required to support all channels with their data presentation requirements, based on screen types and capabilities. Further, the common backend API becomes challenging to maintain, as it must handle the logic of all channel-specific requirements. This can potentially lead to decreased performance. Therefore, there is a need for separate backends for mobile and web. This requirement has led to the development of the backend for frontend (BFF) pattern, also known as headless architecture. This paper examines the business challenges that the BFF pattern helps resolve, making it easier for business readers to comprehend and implement. It also discusses the pros and cons of this pattern, enabling technical and business consultants from information technology (IT) teams to make informed recommendations based on specific needs.

Infosys®
Navigate your next

Contents



Introduction: What is Backend for Frontend?

Backend for frontend (BFF)[1] is a solution of patterns that addresses the exclusive requirements of a wide range of client platforms such as web, mobile, among others. It is used to effectively build composable architectures in software development while preserving the advantages of microservices. It provides seamless and smooth user interaction independent of the front-end application platform[2]. To avoid the need for customizing a single backend for multiple interfaces, it is advisable to create different backend services for distinct frontend applications or interfaces. This approach, initially introduced by Sam Newman[3], can begin with a single server-side BFF per user interface. However, as new requirements emerge, necessitating differentiated handling for each channel, the decision to have separate BFFs can be considered.

Why is BFF Required?

Multiple factors drive the **need for BFF** in the communication cloud, each warranting more exploration.

First, the telecom industry has witnessed **stiff competition** in the past two decades. With numerous players in the market, consistent revenue growth for telecom companies hinges on selling as many services as possible to customers. This has led to worldwide internet penetration and a high demand for Generation Z's favored over-the-top (OTT) packaged as part of telecom offerings. Offered as a **large catalog** to channels in payloads, they become an additional overhead for front-end channels. This underscores the need for a BFF pattern, enabling channels to focus on presentation and user experience, while a backend translates simple channel input into a communication cloud-specific payload.

Secondly, communication service providers (CSPs) must be available across platforms – mobile, self-service portals, partner channels, and wholesaler portals. While having an **omnichannel** approach is critical for CSPs, it may not always be feasible. Each channel offers distinct user journeys and experiences as well as features, and capabilities. Implementing a backend layer customized to each channel allows frontend channels to prioritize user experience. Initially backends can be shared between similar sets of channels, one for web and other for mobile channels.

Further, telecom CSPs always strive to outpace their competitors while enhancing customer self-service. They may offer **competitive pricing** for OTT and other services, with perpetual or limited-time discounts for each service line item. However, considering the sheer volume of **discounts** for each channel, a backend becomes imperative to display the final discounted prices, primarily for the following reasons:

- The process can be time-consuming
- Multiple offers may necessitate middleware for numerous, cumbersome API invocations
- Caching APIs is feasible only to a limited extent when dealing with offer combinations. Moreover, caching itself is a time-consuming endeavor that leads to the deterioration of API performance

To address these challenges, the recommended approach is to calculate and display the discounted price using the BFF pattern. This enables digital commerce APIs or the configure, price, quote (CPQ) pricing engine to operate asynchronously, resulting in a smoother and faster process.

Implementation Considerations

Implementing BFFs calls for planning, design, and consideration of various requirements. We explore two important considerations in BFF implementation: the scope of implementation and the number of **BFFs** required.

Scope of the BFF

When designing the BFF pattern, it is crucial to carefully evaluate the scope of implementation. The key factors for evaluating the scope include:

- Performance: Assess the impact on backend **performance** in the communication cloud and across channels, considering the **capabilities of relevant systems.** For instance, handling synchronous complex pricing and discounts may take a considerable amount of time in the communication cloud. To enhance the user experience, consider caching discounted prices for various combinations closer to the channel
- Platform limits: It is crucial to recognize the **platform limits of the BFF, communication cloud, and channels,** particularly in the context of cloud-based systems
- Time-to-market: Evaluate the time-to-market when comparing various BFF options with differing scopes
- Configurability: Ensure that the **BFF and its scope can be configured** to support the frequent introduction of new devices, plans, and value-added services (VAS) required by finance. Supporting product lifecycle management (PLM) systems is pivotal when assessing the scope of the BFF

Number of BFFs

After determining the scope, evaluate the required number of BFFs. The key considerations are:

- Categorization: Group different channels based on shared features and the type of clients they serve. It is recommended to create a common BFF for each category and assess its compatibility with other factors
- User journey: Ensure a seamless user journey across channels by verifying if the categories are sufficient for having a common API for each channel in the category
- Payload standardization: Confirm that the channels in each category can manage a similar payload for the operation. While the BFF will use a standardized payload for communication cloud interaction, the payload between channels and the BFF should be straightforward and uniform for all channels

High-level Architecture

There are two key ways in which frontend channels interact with backend APIs. These are:

1. Channels using general-purpose APIs

2. Channels using dedicated BFFs

Let us look at each in more detail.

Channels using general-purpose APIs

In this architectural pattern, all channels interact with the same backend APIs, resulting in additional overhead for both the channels and the backend. This approach is suited to specific scenarios and use cases, detailed in the next section. Figure 1 illustrates the architecture for channels using general-purpose APIs.

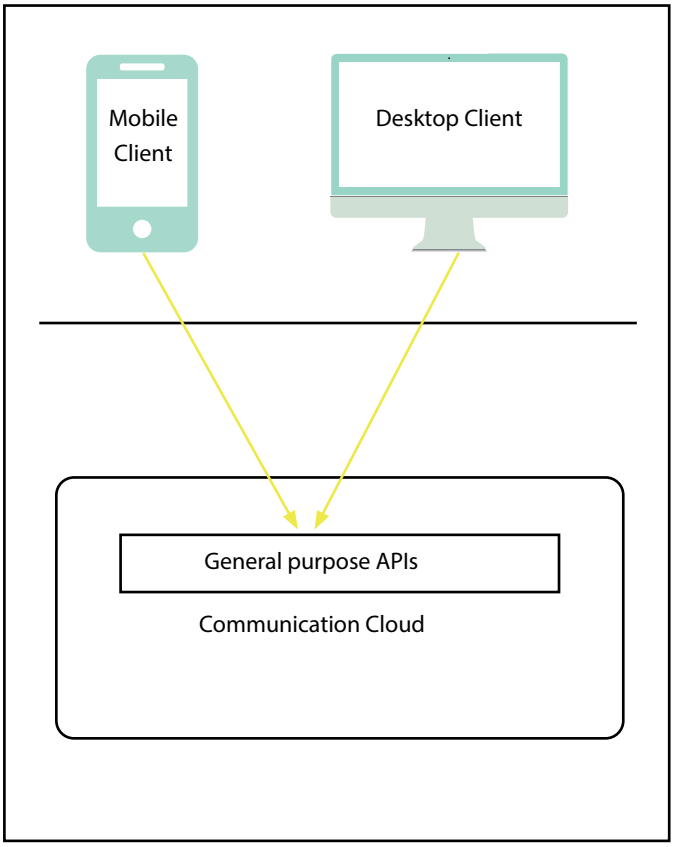


Fig 1: Architecture for channels using general-purpose APIs

Channels using dedicated BFFs

In the BFF architecture pattern, each channel category interacts with its dedicated BFF. The BFFs then interact with common communication cloud APIs. This pattern is explored in the next section, where we discuss the use cases, benefits, and recommendations. Figure 2 depicts the BFF architecture pattern.

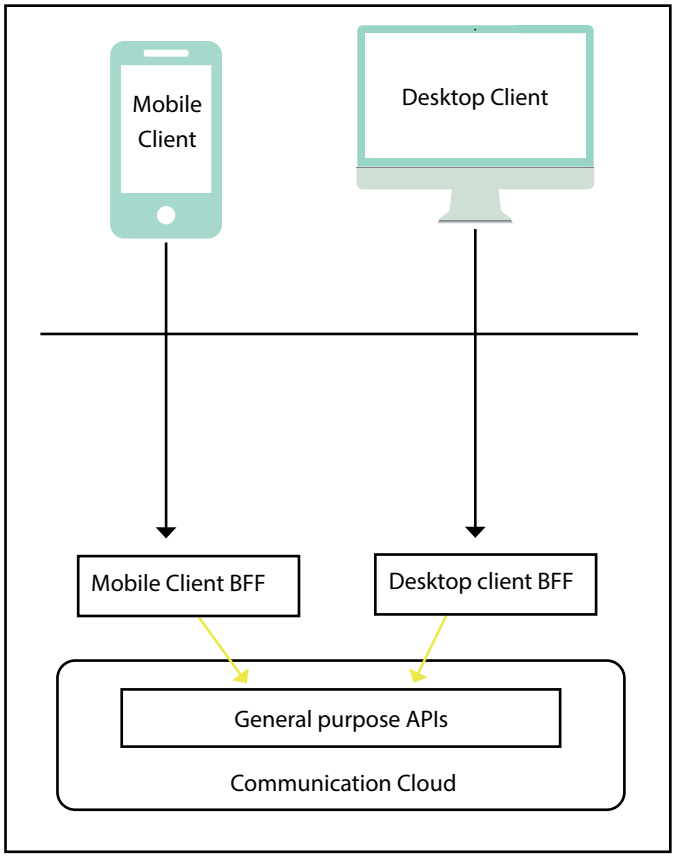


Fig 2: Architecture for channels using dedicated BFFs.

Common Use Cases

The BFF layer offers substantial benefits by streamlining common use cases. It assists with abstracting runtime discount evaluation rules, determining voucher application order and eligibility, and preventing frequent API calls to the communication cloud for runtime evaluation. These tasks, which are potentially complex and time-consuming, can be achieved with greater ease and efficiency by using the BFF layer.

Measuring BFF

When assessing the adoption of the BFF pattern for telecom companies, it is essential to conduct a thorough evaluation of the advantages and disadvantages of this approach to make informed decisions.

Identifying and addressing specific business challenges that CSPs encounter is crucial. A proof of concept (POC) serves as a valuable tool for gauging how the BFF pattern can help mitigate these challenges. For example, a POC can be used to evaluate how the BFF pattern can:

- Increase the number of discounts offered in a competitive pricing landscape
- Enhance the range of offers and services that can be bundled into a single order, thereby boosting revenue
- Improve overall performance and reduce abandonment rates

A typical POC involves developing and testing a prototype of the BFF layer, one of the channels or test clients, and the communication cloud. This approach allows for measuring the effectiveness of the new strategy using key performance indicators (KPIs). It enables an assessment of whether implementing the BFF pattern will effectively resolve existing challenges and achieve desired outcomes. Carefully evaluating the pros and cons of this approach and monitoring its impact on the KPIs can enable informed decisions that lead to increased revenue and customer satisfaction.

Scenarios for General Purpose Backend APIs; Unsuited for BFF Pattern

The BFF architectural pattern involves adding a single backend layer for each channel or category of channels. This can increase maintenance costs and demand additional resources, making it unsuitable for small telecom implementations operating with limited resources and budgets.

New telecom companies offering a limited number of products and services with simple order-level discounts and channels, and capable of managing the required payload, may not need a BFF. In such cases, common communication cloud APIs can be leveraged for all channels.

Telecom companies operating exclusively through their native Salesforce platform channel do not require the BFF architectural pattern. They can rely on unguided or guided order capture journeys depending on whether the users are internal or external.

Companies where performance is not a key parameter, such as those that have orders placed by internal agents on behalf of customers, may delay BFF implementation until they reach a scale where agent performance becomes significant.

Comparing Microservices, BFF, and Headless APIs

Let us compare microservices, BFFs, and headless APIs, as they serve distinct but interconnected roles in backend architecture.
Microservices architecture decomposes a large application into smaller, standalone services that can be deployed independently and communicate with each other through APIs.

BFF is a design pattern that involves creating a dedicated backend service for a specific user interface or client type.
Headless APIs facilitate content retrieval and delivery to various frontend applications or channels such as websites, mobile apps, or Internet of Things (IoT) devices.
While these concepts are related to backend architecture and API design, they are not interchangeable.

Microservices represent a broad architectural approach. BFF and headless APIs are specific design patterns or components within a broader architecture. BFF is often used with microservices to customize backend services for specific frontends.

Headless APIs are general purpose APIs that provides backend services (or data) to presentation layer , front end usually is the content management system.

In summary, microservices provide an architectural approach, whereas BFF and headless APIs are patterns and components used within specific contexts to optimize backend/frontend interactions and content management, respectively.

Collaborative Support from an Expert Partner

Implementing the BFF pattern can boost customer satisfaction, optimize omnichannel experiences, and expand the range of available discounts and products. These advantages have the potential to generate increased revenues, particularly in highly competitive markets with lower compound annual growth rates (CAGRs). Telecom service providers acknowledge that they use the BFF pattern to offer atomic and stackable discounts on each of their products, achieving greater flexibility. It is evident that the pattern improves performance and reduces abandon rates significantly, providing a significant edge in a competitive market.

A reliable and experienced partner is critical for successful BFF pattern implementation. Along with General purpose APIs implementations, Infosys also has implementation expertise in native and non-Native BFFs. With successful BFF pattern implementation in multiple rollouts, we have helped organizations manage complex catalog and discount structures. Our support includes a dedicated telecom lab, an array of solutions, digital platform expertise, proven partner ecosystem, pragmatic operating model, and a team of seasoned technical, functional, and business consultants.

Summary

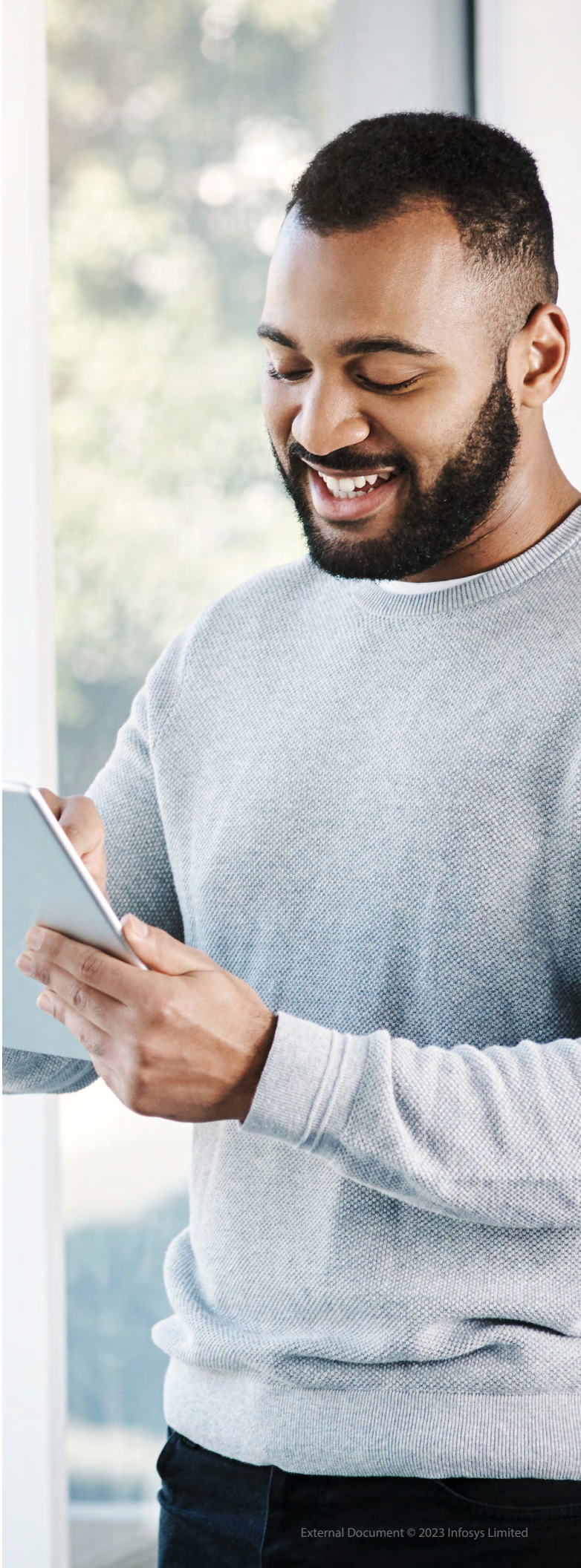
When considering the implementation of a separate BFF between Salesforce communication clouds and channels, a thorough examination of **the benefits and drawbacks of this approach** is vital. The BFF pattern is particularly well-suited for telecom companies that prioritize performance and offer an extensive catalog of products with various discounts. While this paper covers the complexity and need for Telco BFF, this will fit for other industry including energy.

Acknowledgements

We extend our sincere gratitude to the communication cloud subject matter experts and thought leaders from Infosys, Dr Thejasvi Nagaraju, Industry Principal, and Mr Deepak Kumar Jha, Industry Principal, for their invaluable guidance, inputs, and meticulous reviews. We also appreciate the encouragement and feedback from our colleagues at Infosys, Mr Sushil Hinduja, Industry Principal, Mr Pratyush Anand, Senior Technology Architect, and Ms Surbhi Sharma, Senior Consultant, which greatly contributed to the depth and development of this white paper.

Acronyms

API	Application programming interface
BFF	Backend for frontend
CAGR	Compounded annual growth rate
COM	Customer order management
CMS	Content management system
CPQ	Configure price quote
CSPs	Communication service providers
DC	Digital commerce
IOT	Internet of Things
ISP	Internet service provider
KPI	Key performance indicators
MVNO	Mobile virtual network operator
OTT	Over-the-top
PLM	Product lifecycle management
POC	Proof of concept
SOM	Service order management
TMF	Telecommunications Management Forum
VAS	Value-added services



References

1. Backend for Frontend
2. A Deep Dive into the Back-End for Front-End Pattern
3. Pattern: Backends for Frontends
4. Market Insights > Technology > Communication Services - Europe
5. Market Insights > Technology > Communication Services - United Kingdom
6. Market Insights > Technology > Communication Services - United States
7. Guided Competition in Singapore's Telecommunications Industry
8. Singapore Telecom Market Size & Share Analysis - Growth Trends & Forecasts (2023 - 2028)
9. Everything You Need To Know About Headless Commerce
10. Backends for Frontends pattern - Azure Architecture Center

Authors

Mansimar Singh brings over 12 years of experience in designing and developing transformative enterprise solutions in the telecommunications, manufacturing, and retail domains. As a Salesforce Solution Architect, he specializes in digital transformation of order-to-care and customer service landscapes, particularly Salesforce industry-specific platforms like Salesforce Communications Cloud. His focus is on providing platform advisory services and designing solutions that streamline processes with the objective of delivering significant tangible and intangible benefits to businesses.

Hitesh Kumar Vyas, Salesforce Solution Architect, has more than 16 years of IT experience in the telecommunications, insurance, and CRM domains. With a versatile skillset, he adeptly manages multiple projects leveraging his extensive experience in software engineering , IT solution architecture, and business analysis. He has successfully handled large transformation programs in operations support systems (OSS) and business support systems (BSS). His areas of expertise are consulting, solution design, estimation, telecom order management, configure-price-quote (CPQ), designing and deploying fully integrated and automated BSS architecture, establishing end-to-end enterprise architecture IT solutions, and interface definition.

For more information, contact askus@infosys.com

Infosys®
Navigate your next