Forethought

THE ENGINEERING AND CX LEADERS' GUIDE TO

AI Deployment and Headless Implementation

Table of Contents

Intro

Every CX leader knows AI matters. The pressure to deploy, from your board, from your customers, from the sheer volume of tickets flooding your inbox, is real. But knowing AI matters and knowing how to deploy it are two different problems.

CX leaders need automation they can trust to improve the customer experience, not degrade it. Technical teams need control over how it's built, including the code, the data flows, and the security posture. Those needs don't always align, especially when you're staring down the build-versus-buy decision.

If you build in-house, you get total control. Every line of code, every data source, every security setting is yours to manage. You own the UI, the workflows, the tone. For companies with strict compliance requirements or engineering cultures that resist external dependencies, control feels non-negotiable. But you're also committing to months of engineering work before you see any ROI, including endless maintenance to retrain models, monitor for drift, and push updates.

No matter how talented your team is, you'll never match the training data scale of a specialized AI vendor. You're spending R&D cycles building infrastructure instead of improving the customer experience. So teams buy from vendors to avoid that trap. You get fast deployment in weeks, not months, pretrained intelligence, and out-of-the-box models refined across millions of interactions. The vendor handles updates automatically as their platform evolves.

The thing is, traditional vendor solutions come with predefined UIs, workflow structures, and design systems. For engineering teams that value architectural control, standard vendor platforms present a different set of constraints. You gain deployment speed, but you give up the ownership and flexibility that made you consider building in the first place.

This guide walks you through that choice step-by-step. But more importantly, it shows you how to stop thinking about this as a binary decision. You don't have to choose between control and speed. You can choose a vendor that gives you both—one that lets you keep development control while delivering the expertise and framework you'd expect from a mature platform. That's what Headless Forethought does, and that's what this guide will help you evaluate for your own situation.

6 Questions to Decide If You Should Build or Buy

6 Questions to Decide If You Should Build or Buy

The build-versus-buy decision isn't really about technology. It's about three things you're trying to balance: control, resources, and risk.

Control is about owning your data, deciding exactly how the AI behaves, and making sure it aligns with your internal systems.

Resources are about your timeline—how fast you need to show ROI—and your team's capacity to build and maintain AI infrastructure.

Risk is about security requirements, compliance obligations, governance frameworks, and what happens to your reputation if the AI fails in front of customers.

When teams get this decision wrong, it's often because they evaluate these factors in the wrong order. They start by asking what their budget allows or what their engineering team can realistically ship. But those are constraint questions, not strategy questions. If you start there, you end up with a solution that fits your immediate limitations but doesn't actually serve how your business works and creates a complicated design phase.

This section walks through six questions that are designed to surface the trade-offs that matter most to your business. By the end, you'll know whether to build from scratch, buy a vendor platform, or use Headless Forethought, an API or MCP that lets you build custom AI experiences in your own UI while Forethought handles the underlying intelligence, security, and infrastructure.

1. What part of the customer experience is core to our brand?

When choosing whether to build or buy a CX AI solution, what matters is how you interact with customers. Are those interactions part of what makes people choose you, or do those interactions just need to work well, without being core to why the customer is here.

That distinction should weigh on your decision. If the support interactions are a core part of your product, you may want more control over how an AI agent feels, how it flows, and how it integrates into the rest of the customer journey. If the interaction supports your product but isn't the reason people buy, you may need reliability and scale more than you need customization.

Build when your customer experience is the product

If the way customers interact with you is part of what you're selling, you probably need control over the UI, the workflows, and how AI integrates into your product. Consider a travel or hospitality brand. The experience customers have when they reach out to ask questions, rebook a trip, request a refund, or change a reservation could be just as important as the booking itself. If that's true for your business, you might want significant control over the AI solution managing those interactions.

But you also know that building your own AI reasoning and orchestration stack from scratch would take quarters, maybe years. And even then, it might never match the enterprise-grade reliability of a platform that's been refined across millions of interactions.

That's where a solution like Headless could make sense. You build the interface and workflows exactly the way you want them, while relying on Forethought's proven intelligence layer under the hood. You get to "build," but you're building on solid ground instead of starting from zero.

Buy when your customer experience complements the product

If you need reliable, scalable support but don't need to reinvent interaction patterns, buying a full platform might be the better path. You can still configure tone, workflows, and escalation logic without rebuilding the entire stack.

Take a SaaS logistics platform that supports thousands of customers across different time zones, for example. Their customers probably care most about the product itself—specifically, whether it solves their logistics problems. They still care deeply about the support experience, but their competitive edge likely comes from how reliably they deliver help at scale, not from reinventing every interaction pattern. The main goal is to make support frictionless and dependable everywhere.

Here, an out-of-the-box solution like Forethought could be the right choice. You can still tune tone, workflows, and escalation logic to match brand standards, but working within a proven framework. You accept a standardized interaction structure in exchange for faster deployment and lower overhead.

2. Do we want to own UI, intelligence, data, or a mix of all three?

Once you've decided how much of the customer experience you need to own, the next question is which layer that control applies to: user interface (UI), intelligence, or data.

Understanding these three distinct layers helps you determine where you actually need control and where you can delegate.

The **UI layer** is what customers see and interact with—the chat widget, the voice experience, the email layout, the tone and phrasing of replies. This is usually what CX leaders mean when they say "we want control over the experience." Engineering's role here is front-end integration and maintaining brand consistency.

The **intelligence layer** is the brain. It's where intent detection, reasoning, and decision-making happen. It includes the model itself, orchestration between systems, and how data moves in and out. This is where engineering and data teams usually have the biggest stake. They care about security, integration depth, and model performance.

The **data layer** is the systems feeding the AI—your CRMs, help desks, billing systems, knowledge bases, and other sources the AI connects to. Control here means deciding where data lives, what gets shared externally, and how updates flow back into your internal systems.

Most teams don't need to own all three layers. The question is which one matters most to how your business works.

Build when you need to own the UI layer

If customers spend significant time inside your product, the support UI might need to feel like part of that experience, not bolted on top of it.

On a large online education platform, for example, learners might spend hours within the product interface, interacting with dashboards, progress trackers, lesson discussions, or quizzes. If a student gets stuck, they'd probably want AI to appear contextually next to a video or assignment, using the same design language and tone as the course content. The product team might also want to experiment with how AI can help in-app by suggesting lessons or connecting students directly to instructors.

If you have strong front-end engineers but no reason to train your own model from scratch, a solution like Headless could make sense. You build the interface and workflows yourself, embedding Forethought's intelligence through our API and MCP. CX can control tone, escalation paths, and contextual responses, while engineering can control how it integrates into the product. In the background, Forethought would handle reasoning, accuracy, and security.

Buy when intelligence and data management are your main concerns

If your priority is secure, accurate answers delivered quickly, and the UI doesn't need to be custom-built, buying an out-of-the-box platform might be the better choice.

Take a healthcare software vendor serving hospitals, for example. They might have a small but busy support team handling password resets, patient portal issues, and compliance questions for dozens of clients. Forethought chat or email AI agents could work well for this use case. They probably trust Forethought's proven AI intelligence more than they'd trust building their own, and would benefit most from standard integrations.

They might buy because their IT team is focused on HIPAA compliance and uptime, not on front-end development. Every hour spent customizing a UI could be an hour not spent maintaining integrations with electronic health records. They need a solution that's already certified, stable, and proven in regulated settings.

3. Can we lawfully and safely let a vendor process data, and under what controls?

This is the first real constraint check after deciding what you want to own. Before you choose how to deploy AI, you need to classify what kinds of data the system will see: conversation text, personally identifiable information, account details, transaction history, payment data, or health records.

Each class carries different obligations under frameworks like SOC 2, ISO 27001, GDPR, HIPAA, or customer data processing agreements. Some data can be processed by a vendor under contract. Some can't ever leave your boundary, either because of regulatory requirements or internal security policies.

Forethought operates at an enterprise security level. It's SOC 2 Type II and ISO 27001 certified, fully GDPR and CCPA compliant, and supports HIPAA through business associate agreements where needed. All data is encrypted in transit and at rest, with access controlled through SSO, role-based permissions, and audit logs. Customers can also run Forethought within their own infrastructure or behind their firewall to meet internal security and data-residency requirements.

Still, there are scenarios where you may want to keep data in-house—not because a vendor can't meet your standards, but because your architecture doesn't allow certain systems to be exposed at all.

Build with Headless when AI needs to query internal systems that can't be exposed

Consider a financial services firm that wants to automate balance inquiries, refund approvals, or account modifications. These actions might require real-time queries to core banking systems, transaction databases, or fraud-detection platforms that can't be exposed via API to external vendors. The restriction often comes from architectural security policies like internal-only access, no external network paths, and strict change control, rather than regulatory requirements alone.

With a solution like Headless, you could keep sensitive systems behind your firewall. The AI receives minimal context through the API, like customer intent or session ID. Your internal services handle the actual data lookups and return only what's needed. Forethought processes data with context and returns policy-safe responses while you maintain your security architecture with enterprise-grade AI.

Buy when the vendor's certifications already cover your data class

Now consider a pharmacy benefits manager or health insurance provider that supports employers and plan members with coverage questions, formulary details, copay calculations, and prior authorization status. The support teams use policy documents and anonymized member data, but all truly regulated medical data stays locked in their claims and EHR systems.

What the CX layer sees are structured data calls and anonymized details governed by contracts. If the vendor maintains the same security controls and executes a business associate agreement, that information falls within HIPAA protections. The PBM could work with Forethought's out-of-the-box solutions, configure SSO and access logging, and deploy AI agents in email, chat, and voice as hosted solutions without re-architecting or keeping data local. That could make buying the faster, safer, and fully compliant choice.

4. Can we actually run governance to compliance standards, or do we need a vendor to do it?

Questioning your governance capabilities is about asking whether you can monitor AI quality in real-time, test changes before they go live, prove what the AI did through audit logs, enforce policy updates across all channels instantly, and diagnose and fix failures within hours.

That requires dedicated machine learning (ML) engineering teams, automated evaluation pipelines, centralized observability platforms like DataDog or Splunk, version control for prompts and workflows, formal change-management processes, and audit-trail infrastructure.

The real question isn't "do we care about governance?"—everyone does. It's "Do we already operate other systems with this level of rigor?" If you do, integrating a solution like Headless into your existing governance stack could be incremental work. If you don't, building these capabilities from scratch is a six to twelve-month project that pulls focus from your core product.

Build when AI must integrate into mandatory internal governance systems

Consider a large financial services company that gets audited regularly by financial regulators. They might be required to log every system decision in a specific way, with records kept for seven years. Changes to any production system might need formal approvals and staged rollouts. Incidents might need to flow through a specific response process that regulators have already reviewed. All of this gets reported in dashboards that auditors examine during compliance reviews.

If the AI lives in a separate system with its own logs and its own processes, that creates gaps auditors will flag. With a solution like Headless, the AI could plug into the systems they're already required to use.

Logs go where all their other logs go, and changes follow the same approval process as everything else. Incidents trigger the same response workflows, compliance teams see AI alongside everything else they monitor, and the AI agent meets regulatory requirements because it works the same way as every other system they operate.

Buy when you can adopt the vendor's governance framework

Now consider a smaller B2B software company. They're not in a regulated industry, and their engineering team might be thirty people, focused on building product features. They have basic monitoring and simple deployment processes, but nothing formal.

They still need to show enterprise customers that the AI is monitored, that policies get enforced, and that decisions are tracked. But they don't have rules about which specific tools or processes they must use. Building a formal governance infrastructure from scratch would take months and wouldn't help them ship their product faster.

They could use a vendor's governance tools instead. The vendor provides audit logs, access controls, policy management, and compliance reporting to meet standard enterprise requirements. That's usually enough, because there's no regulatory mandate requiring them to build and run these systems themselves.

5. Can we scale to new channels without degrading the customer experience?

Most companies start with one channel, like chat on the website, and then expand when customers want to reach them elsewhere: email, phone, SMS, or in-app. The challenge isn't just making sure answers stay consistent—both building and buying could solve that with shared intelligence, one policy layer, and unified analytics.

The real question is whether you need custom UI or non-standard channel implementations, or whether pre-built channels work for you. Standard channels like chat widgets, voice IVR, and email triage work for most companies. They're fast to deploy and maintain, but if support needs to be deeply embedded in your product, or you need channels the platform doesn't support, like kiosks, in-car systems, wearables, or custom IVR, you may need to build the UI layer yourself.

Build when you need custom UI per channel or non-standard channel implementations

Consider a ride-share marketplace where support appears in three places: the rider app for in-transaction chat, the driver app for earnings and policy questions, and a phone line for urgent safety issues. A standard chat widget wouldn't work because rider chat needs to be embedded next to trip details with full ride context, including pickup location, driver info, and fare breakdown. The driver app has an entirely different UI focused on earnings, policy compliance, and vehicle requirements, and the phone experience needs custom IVR flows that route safety escalations to human agents.

Each channel might need a purpose-built UI that feels native to that product experience, but they'd still need guaranteed consistency, including the same policies, the same knowledge base, and the same escalation logic across the rider app, the driver app, and the phone.

With a solution like Headless, they could build custom UI components for rider chat, driver chat, and IVR flows. All three would call the same Forethought intelligence via API. The CX team could update policies once, and the changes would automatically propagate to all three channels. An analytics dashboard could show performance across all channels

despite different UIs, and they'd get full control over how support appears in each product, with guaranteed consistency because one brain powers all channels.

Buy when pre-built channel implementations meet your needs

Now consider an e-commerce company. Customer demand for phone support and email inquiries is growing, and standard implementations could work fine for them—chat widget on product pages and checkout, voice IVR for order status and returns, email triage for non-urgent inquiries. They don't need support embedded in unique product experiences. They just need reliable, consistent answers across standard channels.

They could deploy pre-built chat, voice, and email solutions, configure branding like colors and logo, and use standard UI components. One knowledge base and one policy engine would power all three channels, and if they wanted to add SMS next quarter, it could be a configuration setting rather than a development project.

6. How fast do we need to deploy, and what's the lifetime cost of delay?

You've assessed what you need to own and whether you have the infrastructure to support it. Now the question is timing—building custom implementations takes time. An out-of-the-box platform, on the other hand, deploys in weeks. Headless deploys based on your team resourcing , which is faster than building from scratch, but not instant.

If you're scaling fast, under pressure to reduce support costs, or facing customer experience issues right now, speed might matter more than perfect customization.

Build when you can absorb the timeline and have specific requirements that justify the investment

Consider an enterprise software company planning an eighteen-month CX transformation. They might be rolling out a new product experience across multiple customer segments, and support needs to be tightly integrated into the latest product UI rather than bolted on afterward. If they have a six to nine-month timeline before launch, building a custom integration with Headless could fit their roadmap.

Custom requirements might justify the investment, such as multi-product support surfaces, role-based experiences, and complex workflow routing. The engineering team could build during the product development cycle and launch an integrated experience when the product goes live.

Buy when you need results now, and speed outweighs customization

Now consider a high-growth SaaS company doubling its customer base every quarter. Support ticket volume might be growing by forty percent quarter-over-quarter. They're hiring support agents as fast as they can, but they can't keep pace with growth. They need deflection and automation within 60 days, not 6 months.

An out-of-the-box implementation could meet their needs because they need a solution live fast. They could deploy in four to six weeks, start deflecting tickets immediately, and iterate on tone and workflows while the system is already running.

The choice to build or buy comes down to control vs. speed

The build-versus-buy decision comes down to one fundamental trade-off: control versus speed.

If owning the customer experience is essential to your product—if support needs to feel native to your interface, if you have systems that can't be exposed externally, if you must integrate AI into mandatory internal infrastructure—then the months it takes to build with Headless could be worth it. The alternative, forcing your product into a vendor's framework, might create a worse customer experience than waiting.

But if your differentiation lives in your core product, not in how support looks or works, then every month spent building custom UI is a month of lost deflection, rising costs, and delayed ROI. Standard implementations are fast, proven, and often good enough. "Good enough" beats "perfect in six months."

Headless is the Best of Both “Build” and “Buy”

Headless is the Best of Both "Build" and "Buy"

If you've worked through the six questions and landed somewhere in the middle, you want proven AI intelligence and enterprise security without building from scratch. But, you also need control over the UI, embedded experiences, or integration with internal systems that a standard platform can't reach. That's what Headless is for.

Headless is Forethought's API and MCP solution. It gives you everything the platform provides—AI intelligence, security, orchestration—but lets you build the customer-facing experience yourself. You're buying the complex parts: model training, compliance, and ongoing maintenance. You're building the parts that differentiate your product: UI, workflows, channel implementations.

It's not "build versus buy." It's "buy the foundation, build the experience."

Headless gives you both control and speed

The traditional trade-off is choosing between control (build from scratch but take on months of work) or speed (choose a vendor solution but accept constraints). Headless gives you the best of both worlds with three developer tools: SDK, MCP, and API.

These tools let you build custom experiences in weeks while leveraging Forethought's proven AI intelligence. You write the front-end code, and Forethought handles the AI brain.

You get control

You can build your own stack using your frameworks like React, Vue, or Next.js, and design your own UI without pre-built widgets. You choose which layers you own—like the UI and workflow logic—while delegating intelligence to Forethought, and you can run inside your infrastructure while deploying behind your firewall if needed, controlling data flows and setting network boundaries.

You get speed

The SDK lets you drop AI agents into any UI with just a few lines of Python or JavaScript code. CX teams can still update workflows independently without creating an engineering bottleneck, and our MCP means you define capabilities once. Forethought auto-discovers

and routes, without manual endpoint wiring. The API is a clean REST API with streaming responses in text, HTML, or markdown so that you can integrate AI anywhere in your system.

You also get prebuilt intelligence from models trained on over a billion interactions per month. Headless doesn't require ML ops hiring, retraining, or drift monitoring. It's automatically updated, and can be implemented in weeks, not quarters.

Headless handles the intelligence while you own the stack

Forethought runs the AI models that power customer interactions—understanding what customers are asking, generating accurate responses, and deciding when to escalate to humans. These models are trained on over a billion interactions monthly across Forethought's entire customer base, a scale no individual company can replicate. Our models improve automatically as Forethought processes more data, so your AI gets smarter without your team doing anything, and also handles multi-channel orchestration, meaning one intelligence layer works consistently across chat, email, voice, and SMS.

What you own is everything customers see and interact with—the UI, how support appears in your product, your design system, and branding. You own your tech stack, which means you choose your frameworks, control your deployment, and integrate with your internal systems. You also own your integration architecture, deciding what data the AI can access, how it connects to your backend, and where it fits in your workflows.

This division works because you can't beat a specialized AI vendor. They have more training data and ML expertise than you'll ever build in-house. But they can't beat you at knowing your product and customers. Headless lets you focus engineering time on what differentiates your customer experience while buying the hard infrastructure work from experts who've already solved it at scale.

Headless comes with enterprise-level security:

Headless provides SOC 2 Type II- and ISO 27001-certified infrastructure, audited annually and maintained by Forethought. Full GDPR and CCPA compliance is built into the platform, and it's HIPAA-ready through business associate agreements when needed. All data is encrypted in transit and at rest by default, with SSO integration, role-based access control, and comprehensive audit logging out of the box.

You also get deployment flexibility. For most companies, you can run Forethought in their secure cloud infrastructure with standard enterprise controls. For companies with stricter requirements, you can deploy behind your own firewall or within your VPC to meet internal security policies and data residency requirements. You maintain control over data flows and network boundaries while leveraging Forethought's certified AI.

Building and maintaining enterprise security certifications in-house takes six to twelve months and dedicated compliance staff. Forethought has already done this work and continues to maintain it as our customer base grows. You get security infrastructure that would cost hundreds of thousands of dollars to build and certify yourself, without the ongoing maintenance burden, and your security and compliance teams can validate Forethought's certifications rather than building everything from scratch.

Headless is one brain across every channel

Forethought's intelligence layer powers every channel from consistent sources, like your knowledge base, policies, or set of workflows. When your CX team updates a return policy or fixes a knowledge article, it can be applied to chat, voice, email, and SMS without separate deployments per channel. The AI reasoning stays consistent, so a customer asking about refunds gets the same answer whether they're chatting, calling, or emailing.

Headless delivers this by letting you build custom UI for each channel. All of them call the same Forethought intelligence via API to understand intent, retrieve the right information, follow policies, and decide when to escalate. Your UI layer stays flexible while the intelligence stays consistent underneath.

This matters because adding a new channel becomes a UI project, not an AI project. You're not rebuilding logic or retraining models. CX can still manage a single set of policies and workflows, rather than maintaining separate systems per channel, while engineering builds once and reuses intelligence across every surface without duplicating infrastructure.

Best Practices to Implement Headless AI

Best Practices to Implement Headless AI

Most implementation guides walk you through API keys, authentication, and endpoint testing, but the hard part is deciding where to deploy AI first and what to automate before you've proven it works.

The companies that show ROI in weeks make different choices upfront than the ones still tweaking configurations months later. The difference is knowing where AI creates the most value fastest, and where it's likely to stall. These five practices come from companies that successfully deployed Headless, and they surface the patterns that tend to work.

Embed AI where your customers already are

The instinct when deploying AI is to build a help center or add a chat widget in the corner of your screen. But that creates a separate destination that customers have to find and decide to use, but most won't. The companies that see higher deflection rates put AI directly into the workflows where customers get stuck.

One customer success platform embedded Solve into their product for technical support tasks. When a customer hit a password reset issue or couldn't access their account, the AI appeared right there, in context. They didn't have to hunt for a help button or explain their situation from scratch—the AI agent already knew what they were trying to do because it lived inside that workflow.

Before you deploy, map where your customers spend time in your product. Consider places like transaction pages, checkout flows, dashboards,and settings screens. Then look for friction points—after placing an order, during onboarding, when trying to change account settings. Those are the moments when customers need help, and that's where AI should appear. If you use your existing UI patterns, the less talking to an AI agent will feel like "getting help," and the more it will feel like the product is just working.

Start with high-volume transactional tasks

When you're choosing what to automate first, the obvious answer is the high-volume tasks that eat up the most agent time. But not all high-volume tasks are good starting points.

Start with the ones that follow clear policies and don't require judgment, like password resets, order status checks, and refund requests that fit a simple rule like "within thirty days and unused." These tasks are predictable and easy to measure. If something goes wrong, the risk is contained, and you can prove deflection quickly.

One workflow automation company started by using Headless to build an agent that answered questions from their knowledge base. Once that worked, they expanded into actions: resetting two-factor authentication, updating billing records, and verifying customer data before system changes. They went from answering questions to actually solving problems, and deflection rates jumped because customers got their issue resolved without waiting for a human.

Let CX own policies while engineering owns the stack

AI implementations can create a dependency problem. CX needs to update a refund policy or change the tone of a response, which shouldn't require engineering, because then engineering gets flooded with tickets for changes that have nothing to do with code. CX waits days or weeks for updates that should take minutes, and both teams get slower.

Headless fixes this by drawing a clear line. Engineering builds the structure—the UI, the API connections, the authentication, and how data flows. Once it's done, CX takes over the operations and can update policies, adjust tone, change escalation rules, and fix knowledge articles. All of that happens in Forethought's admin dashboard without touching code.

One customer success platform used Headless to embed AI agents into their in-app knowledge center. The platform integrated with Slack and pulled from community content to help customers troubleshoot issues and learn best practices.

It worked well because CX could update what changed frequently. When a policy was updated, they updated it in the dashboard, and it applied across every channel instantly without engineering intervention.

Maintain one source of truth across all touchpoints

If customers get different answers depending on whether they email, chat, or call, they stop trusting your support. And if you're updating policies in multiple places, something will get out of sync.

One education platform had two AI systems: a tutor for academic questions and a bot for support. Students didn't always know which one to use, and there was a real risk they'd get conflicting answers. So they used Headless to embed a support AI agent inside the tutoring interface. Now, when a student asks a support question, it routes through Forethought even though it looks like it's part of the tutor. One system handles all support questions, so students get consistent answers and everything is tracked in one place.

Measure what matters to prove ROI

Don't just measure overall AI performance. Measure each use case you automate separately. That tells you which types of automation work and which don't, and provides proof to justify expanding to more complicated problems.

One workflow automation company achieved 78% deflection in the first two weeks with Headless for knowledge retrieval. That fast win proved that the simple automation worked, which justified moving resources to more complex transactional workflows, such as two-factor authentication resets and billing actions. They measured time-to-resolution for each use case separately, showing that eliminating manual steps for specific tasks sped up customer resolution and reduced ticket volume. That granular measurement gave them the proof they needed to keep expanding.

Proven AI, Deployed Your Way

Proven AI, Deployed Your Way

Forethought gives you two ways to deploy: Headless or the Forethought platform. Both use the same AI intelligence trained on over a billion interactions monthly. Both deliver enterprise security, multi-channel consistency, and measurable ROI through deflection rates and reduced resolution time. The choice is about which deployment model fits your product, your team, and your timeline.

With Forethought, you get pre-built channels like chat, voice, Slack, and email that you configure to match your brand and workflows. You can deploy in weeks, start deflecting tickets immediately, and let CX teams manage policies while engineering handles authentication and data access.

With Headless, you get full control over the UI, your frameworks, and how AI integrates into your product. You build custom experiences per channel while leveraging Forethought's intelligence through the API and MCP. Engineering owns the UI layer, and CX owns the policies.

The six questions in this guide surface your constraints: what you need to own, what data you're working with, whether you have governance infrastructure in place, how many channels you're supporting, and how fast you need to deploy. Your answers point you toward the right solution. If you're still uncertain, start with whichever approach matches your most critical constraint, whether that's speed or control.

Either way, you're building on the same proven foundation. The companies that succeed with AI aren't the ones who build everything from scratch or blindly accept vendor constraints. They're the ones who understand exactly what they need to own and what they can delegate.

Want to see how Headless works for your specific situation? Talk to Forethought's team about your constraints and deployment options.

Request A Demo