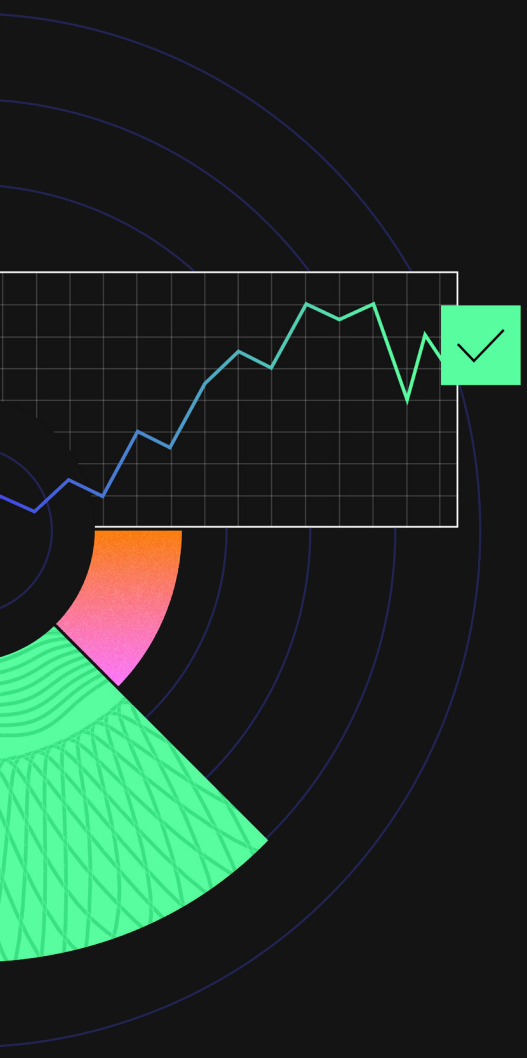
Temporal

RISK AND RELIABILITY

How Financial Services and FinTech Companies Reduce Risk with Temporal

Table of Contents



14.3%

increase in global risk and security spending forecast for 2024 compared to 2023

Risk management isn't just another task to check off — it's the foundation of any forward-thinking business strategy. Whether guiding business decisions or developing in the trenches, you understand the obstacles that can compromise your organization's stability: outdated systems, outages, and regulatory violations that affect both revenue and reputation.

As the market evolves, so does regulatory scrutiny, with 75% of the world's population expected to have their data governed by modern privacy regulations by 2025. The fallout from high-profile outages, like the 2024 CrowdStrike incident, which cost Fortune 500 companies around $5.4 billion, is a stark reminder of what's at stake. Businesses are taking heed. Gartner even forecasted a 14.3% increase in global risk and security spending in 2024 compared to 2023, with 60% of survey respondents noting an increase in budget, and annual adjustments accounting for 20% of cases. Increased risk and digital transformation were cited as direct reasons for these budget increases.

For engineering leaders and developers alike, risk isn't just about managing your tech stack — it's about protecting your organization's future. Temporal offers a proactive approach to improving resilience and reducing risk across critical systems, setting your team up to bolster reliability where it matters most.

38%

of CIOs listed transforming legacy business processes as a top priority

Addressing the Risks of Unreliable Systems and Legacy Infrastructure

The first step to reducing risk is understanding the limitations of your current systems. In 2023, 38% of CIOs listed transforming legacy business processes as a top priority — and for good reason. While familiar, legacy systems create significant challenges that stifle innovation and increase risk. Here's how to start reducing risk in your infrastrcture:

Modernize Monoliths

Reducing risk begins with moving away from monolithic architectures. By nature, monoliths are difficult to maintain and accumulate technical debt, making them susceptible to outages and security vulnerabilities. Their complex structures not only hinder agility and create bottlenecks that slow down your team's response to market needs.

The limitations of monolithic systems extend beyond immediate operational risks. As organizations grow and innovate, these systems require extensive reworking just to implement minor changes, creating frustration in developers and, ultimately, discouraging a culture of continuous improvement.

Scaling also becomes a significant hurdle. As demands increase, monolithic architectures struggle to keep pace, affecting performance and risking long-term success. Shifting to flexible, modular architectures enables organizations to support sustained growth.

Divest from Choreography-Based Event-Driven Architecture

Today, many financial service organizations have transitioned from monolithic architecture to event-driven architecture (EDA). Often choreography-based, this type of EDA can introduce significant complexity and potential for error. Improving its reliability is essential to minimize risk. One key challenge is the high failure likelihood associated with the complex, interconnected components of this architecture. As each component adds complexity, the system becomes more prone to failures.

Choreography-based EDA, in particular, poses unique challenges. Unlike orchestration, where a central coordinator manages the workflow execution, choreography relies on individual services to autonomously react to events and trigger subsequent actions. This decentralized approach can make it difficult to manage dependencies and lead to circumstances where services behave unpredictably.

Managing event-driven systems at scale introduces additional vulnerabilities. Logic is scattered across multiple services, making it difficult for developers to reason about different events and their interactions. The potential for coding errors rises, compromising system reliability. This type of work is difficult to get right even with relatively small systems, so it's important to move away from this architecture where possible.

Beyond this, tracking transactions and events across a growing system becomes challenging without proper visibility. Troubleshooting an issue with a single transaction can be a long, manual process, which requires sorting through logs belonging to multiple different services. As a result, the lack of transparency hinders your operational efficiency, only making it harder to resolve issues as they arise.

Improving the reliability of event-driven architectures is critical for businesses aiming to scale successfully without sacrificing stability. Divesting from choreography-based EDA re-centralizes control and reduces the risk of failure, making recovery from potential issues and overall coordination easier.

Leave BPMN and Legacy Orchestration Tools Behind

When looking for viable solutions, many organizations turn to Business Process Modeling Notation (BPMN) and legacy orchestration tools to address the challenges of event-driven architecture, specifically choreography-based EDA. While traditional tools like BPMN and legacy orchestration frameworks have long been industry standards, they now present risks to businesses. These tools, though comfortable due to their familiarity, are difficult to scale and create performance issues that negatively affect both productivity and customer experience.

One of the main drawbacks of these systems is that they are not code-first. They rely on rigid, non-programmatic models, limiting developers' ability to build, test, and innovate. Complexity grows exponentially with BPMN-based systems, where proprietary interfaces conceal business logic, making it challenging for developers to fully understand system behavior. This opacity can lead to unexpected failures, adding to downtime and reliability risks.

How Temporal Reduces Operational Risk and Ensures System Reliability

For companies looking for a new approach, Temporal offers scalability, flexibility, and control that legacy tools simply cannot match. Moving away from monoliths and outdated frameworks prepares businesses with a foundation of resilience and innovation built to meet the demands of modern consumers.

Temporal enables developers to code complex, resilient workflows that handle failures with ease. With Temporal, developers can use their preferred programming languages and tools, cutting down on learning curves and increasing productivity.

Created by developers for developers, Temporal addresses the challenges of delivering reliable, scalable systems. By simplifying the creation of complex workflows, Temporal minimizes the risks and inefficiencies often inherent in modernization and distributed systems.

Accelerate Modernization of Monoliths

Financial institutions that cling to legacy systems face heightened risks. Modernization is essential to support innovation, but it's often a complex and costly endeavor. Temporal accelerates this journey by simplifying key aspects of modernization projects, helping financial service organizations move away from outdated systems.

Temporal enables developers to focus on building code that supports core business logic, allowing them to manage intricate workflows more effectively and implement new features faster. This benefit is especially relevant for mobile-to-core modernization efforts, which often involve connecting disparate services and updating monolithic architectures. Temporal abstracts the complexity of handling failures, orchestrating distributed processes, and state management.

> “Over the course of a single weekend, we achieved with Temporal what had eluded us for an entire year.”
>
> — ANZ Team

With Temporal, ANZ achieved project delivery

8x

faster than before.

CASE STUDY

One example of Temporal’s impact in this realm comes from ANZ, one of the largest banks in Australia and New Zealand. ANZ has made strides in improving their level of service by building customer-centric applications. They recently rolled out ANZ Plus, an app designed to help customers manage their expenses, budgets, and access expert support.

The issue they had was building the home loan origination system for ANZ Plus. It required a complete re-architecture of legacy systems, a challenge the team faced for over 18 months before turning to Temporal. Within six weeks of adopting Temporal, ANZ had their home loan origination system in production, achieving project delivery at eight times their previous speed.

According to the team at ANZ, “Over the course of a single weekend, we achieved with Temporal what had eluded us for an entire year.”

Temporal addressed several critical challenges for ANZ’s modernization efforts:

1. **Error-handling abstraction:** Developers could focus on business logic without needing to manage error-handling manually.
2. **Automated compensations and state management:** Temporal ensured all processes were completed in the correct order or rolled back automatically.
3. **Smooth integration with third-party APIs:** Temporal’s support for human-in-the-loop interactions and configurable retry policies allowed seamless integration with systems like Salesforce, preventing process timeouts.

By allowing ANZ to streamline complex workflows, reduce manual intervention, and improve project delivery speed, Temporal facilitated a scalable, domain-driven design for the team. This freed up time for ANZ to focus on delivering great new services while reducing complexity.

Replace Choreography-Based EDA and Legacy Orchestration Tools to Reduce Complexity

Temporal provides a solution to the challenges of both choreography-based EDA and legacy BPMN systems, offering a way to streamline complex, distributed workflows.

One of the key advantages of Temporal is its ability to centralize workflow logic while enabling scalability and flexibility. By providing a unified place for workflow execution, Temporal reduces the development and maintenance complexity often associated with scattered event-driven logic.

Developers can work with the entire workflow in one place, simplifying troubleshooting and reducing the risk of coding errors that come with managing distributed services independently.

Temporal also helps bypass the limitations of legacy BPMN tools. These legacy tools often rely on visual representations that can become difficult to manage as workflows grow in complexity and require proprietary domain-specific languages (DSLs) or inflexible languages like XML, JSON, or YAML to define them. Temporal uses code-based workflow definitions in Java, Typescript, Go, Python, .NET, and PHP, allowing developers to use the programming languages and paradigms they know and love.

This approach improves maintainability and enables more efficient version control and updates. Additionally, Temporal's state management capabilities ensure that workflows are fault-tolerant and recoverable, significantly enhancing system reliability while simplifying errors and retry handling across distributed services.

By adopting Temporal, financial service organizations replace risky choreography-based EDA and outmoded BPMN tools with a developer-friendly alternative.

Customers often report up to 50% less downtime and significant boosts in reliability with Temporal's fault tolerance.

Build Resiliency and Reliability into Critical Processes

In financial services, system outages and instability have costly effects, from lost revenue to customer frustration or even a damaged brand reputation. For crucial activities like processing rent payments, mortgage installments, or other high-stakes transactions, ensuring system resilience is essential.

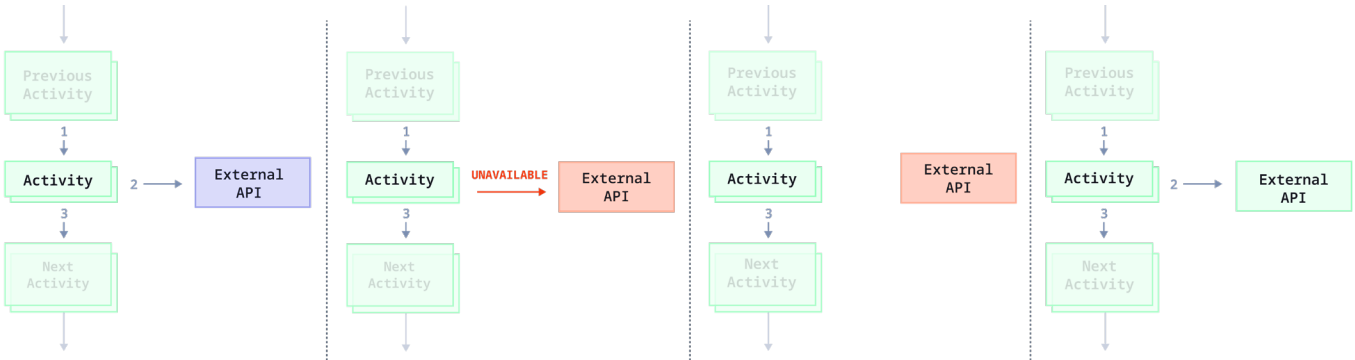
Temporal helps financial institutions achieve near-continuous uptime by streamlining how developers recover and manage systems when issues arise. Customers often report up to 50% less downtime and significant boosts in reliability. Temporal's fault tolerance has two critical capabilities: **1) the automatic handling of retries** and **2) automatic recovery from the point of failure**. These ensure that critical workflows run smoothly, even during disruptions.

In the following sections, we'll deep dive into these two critical capabilities.

Deep Dive: Automatic Retries

In the example below, you can see how an automated retry policy can safeguard recurring transactions:

The diagram illustrates how Temporal manages retries and timeouts. When an external dependency like an API is temporarily unavailable, Temporal can automatically retry the request, ensuring that transactions, like payments, do not get lost in the system. Retry behavior, such as the maximum number of retry attempts, can be customized by developers to suit business requirements.



In normal conditions, the activity will send a request to the external API and receive a response. The activity completes and the workflow will continue.

If the external API is unavailable the activity will fail. The workflow won't make further progress until the activity can complete.

Temporal will retry the activity as per the activity's retry policy. By default, Temporal will retry until the activity completes successfully.

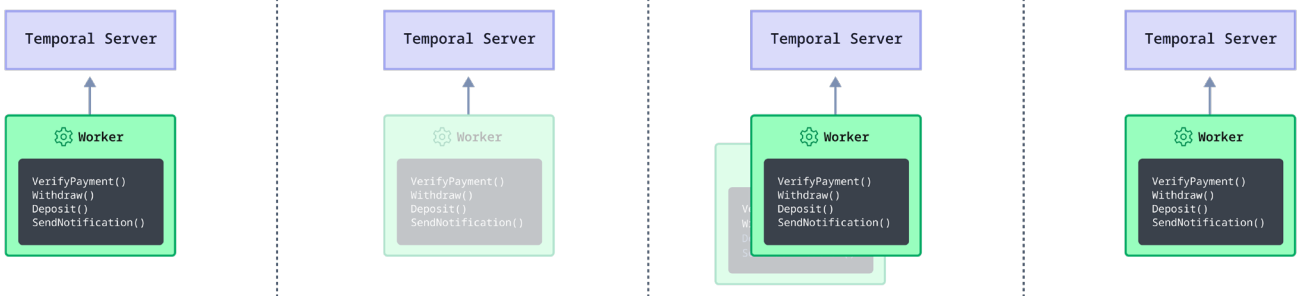
Once the external API is available the activity completes successfully and the workflow progresses.

This capability to customize retry logic allows developers to create fault-tolerant systems that are resilient by design.

Deep Dive: Automatic Recovery from the Point of Failure

Temporal's capacity to maintain the entire running state of a workflow means that if a failure occurs, Temporal will resume a workflow from where it left off. This protects data integrity and ensures uninterrupted service, even in the face of unexpected issues.

In the following diagram, a Worker has already processed two of the four steps in a Workflow:



A worker has processed two of four steps in a workflow.

The worker unexpectedly crashes.

Temporal restores the workflow on another worker. The workflow continues from the point of failure.

The workflow completes execution

For example, this could represent a process that involves intaking a request to schedule a money transfer and then sending the money. If an external factor causes the worker to crash in the next phase, without Temporal, the workflow might need to start over, risking a double transfer and leading to user dissatisfaction

.

With Temporal, the Workflow is restored on another Worker and continues from the point of failure, making the transaction seamless and meeting user expectations. Although it is possible to configure this type of restoration without Temporal, the process is difficult, time-consuming, and often error-prone.

With Temporal, you can build dependable workflows, enhancing system reliability for all of your mission-critical processes with ease.

In Conclusion

In financial services, resilience isn't just a safeguard, it's a competitive advantage. Consumer expectations are higher than ever and only increasing. Financial institutions need systems that don't just keep up but stay ahead. Outages, delays, and failures are costly and damage revenue, customer trust, and compliance.

Temporal offers financial services teams an opportunity to rewrite the rules on reliability. It's more than a platform; it's a new framework for building fault-tolerant, always-available systems that support critical operations without fail. It enables financial services organizations to modernize risky monoliths, move away from complex choreography-based EDA, and streamline development.

By using Temporal, you gain a tool that not only reduces operational risk but powers innovation. Set a new bar for reliability. Start today with a free trial of Temporal Cloud.

Temporal

JOIN OUR COMMUNITY SLACK

EXPLORE PROJECT-BASED TUTORIALS

AND DOCS