Data Modernization

Global Energy Technology Leader

How We Replaced a Critical Offline Sync Platform for a Global Energy Leader Without Missing a Beat

Our client is a global energy technology leader, operating in over 100 countries with a workforce exceeding 100,000 employees. Their teams work in some of the harshest and most remote environments on earth, from offshore rigs to isolated field locations where connectivity is unreliable or non-existent.

To keep operations running, they relied on a platform that allowed employees to work offline for extended periods of up to 30 days before seamlessly synchronizing data once reconnected. This capability was not a luxury. It was mission critical.

But the platform they depended on was being phased out. With end-of-life approaching, they needed a replacement that could match its capabilities and scale, without disrupting global operations. That's when they turned to us.

Utilized Technology Stack:

Cloud: Atlas Stream Processing, Microsoft Azure Event Hubs, Microsoft Azure SignalR, Kubernetes.

Backend database: MongoDB Atlas Database

Backend tech stack: ASP.NET Core

Frontend database: Realm

Frontend tech stack: Electron, Node.js

Monitoring: Microsoft Azure Monitor, Microsoft Azure Application Insights

Review of the Challenges

To keep operations running, they relied on Atlas Device Sync that allowed employees to work offline for extended periods of up to 30 days before seamlessly synchronizing data once reconnected. This capability was not a luxury. It was mission critical.

Atlas Device SDKs were being phased out, but Realm database remained available. With end-of-life approaching, they needed a replacement for the synchronization mechanism that could match its capabilities and scale, without disrupting global operations. That's when they turned to us.

This was not a simple replacement project. The system needed to support tens of thousands of devices operating across the globe, each with the ability to function independently when offline and synchronize reliably when reconnected.

At its core, the challenge was to recreate and improve real time, bi-directional data synchronization while also handling delayed updates, historical data recovery, and conflict resolution. The system needed to track every change, distribute updates intelligently, and ensure nothing was lost, no matter how long a device had been offline.

Alongside the technical complexity came operational challenges. Requirements were initially unclear and continued to evolve throughout the project. There was no dedicated product owner to guide scope, and feedback during early development was limited. Quality assurance was introduced late, and the delivery team had to work within a rigid and strict delivery process that restricted flexibility.

In short, the environment demanded both technical precision and the ability to adapt quickly under pressure.

Our Approach

We began with a series of in-depth discovery sessions to understand the existing system, identify gaps, and define the scope as clearly as possible. From this, we developed architecture designs and a structured backlog that guided delivery.

Development followed the client's Scrum methodology, with daily communication, regular updates, and continuous collaboration between teams. Where clarity was lacking, we worked closely with stakeholders to define requirements in real time. Open communication channels allowed us to resolve issues quickly and keep progress on track.

Despite constraints, the team maintained momentum by focusing on delivery, clarity, and collaboration at every stage.

During the onboarding phase, several discovery sessions were scheduled to try

to clarify the scope, understand existing solution, tech stack and limitations. Based on those sessions, the Team prepared and presented to the client several architecture diagrams and the backlog of items to work on.

Our Solution:

We didn't just replace the existing platform. We re-engineered how data moves across the entire system.
At a high level, the solution is built around a microservices architecture, with a clear separation between data capture, processing, and delivery.
When changes happen in the central MongoDB database, they are picked up using Atlas Stream Processing. These changes are then streamed through Azure Event Hubs, which acts as the backbone for handling high volumes of data reliably and at scale.
From there, updates are distributed in two ways. For real-time communication, Azure SignalR is used to push live updates to connected clients. At the same time, changes are stored and structured so that devices which have been offline can retrieve them later in the correct sequence.
On the client side, applications maintain a local database using Realm. This allows users to continue working without connectivity. Any changes made locally are captured, queued, and sent back upstream once the device reconnects.
To ensure consistency, the system separates historical sync from live updates. When a device comes back online, it first downloads and applies all missed changes. During this process, any new incoming updates are temporarily buffered. Once the device is fully up to date, live changes are applied in real time. This avoids one of the most common failure points in distributed systems: data arriving out of order.
The use of microservices means each part of this pipeline can scale independently. If thousands of devices reconnect at once, the event streaming and processing layers can handle the surge without impacting other parts of the system.
Just as importantly, the entire solution is containerized and cloud-agnostic. While it integrates with Azure services, these components can be swapped for alternatives such as Kafka or other messaging platforms with minimal change.

The result is a system that is not only reliable under real-world conditions, but also flexible enough to evolve as the client's needs change.

Subsequent outcomes:

The final solution gave the client something they did not have before: full control.
By removing reliance on third-party platforms, they eliminated the risk of future deprecation and gained the freedom to evolve the system on their own terms. The architecture is cloud-agnostic, meaning components can be replaced or adapted with minimal effort, ensuring long-term flexibility.
The system integrates seamlessly with existing tools such as authentication and monitoring, and its modular nature allows new features to be introduced without disrupting the wider platform.
It is scalable, resilient, and built for the realities of global operations.

Result:

The platform now supports tens of thousands of devices worldwide, enabling users to work offline for extended periods while maintaining data integrity. Synchronization occurs in near real-time when connectivity is restored, ensuring that operations remain smooth and uninterrupted.
The client now benefits from improved scalability, greater cost control, and a future-proof system that can evolve alongside their business.

Client feedback:

The client's development team responded positively to both the process and the outcome. They highlighted the high quality of the code, the speed at which the solution was delivered, and the clarity of the knowledge transfer provided.
They also recognized the long-term value of the architecture, noting that the modular design makes it easy to introduce new features without major changes to the existing system.

We replaced a mission-critical offline sync platform for a global energy leader, without disrupting operations across tens of thousands of devices worldwide.