Bullish

NEWS RELEASE

Bullish Custody: Smart contracts in action – innovations and insights

2024-10-31

TLDR: In this blog post, we explore Bullish's experience implementing smart contracts within our private Taurus blockchain. We highlight how smart contracts have enhanced security, automation, and data integrity across critical processes. Additionally, we address challenges such as adapting to evolving business requirements, ensuring compliance with regulatory standards, and integrating contracts with distributed systems. We also share key innovations, such as Protobuf-based interfaces and synchronized executions, along with actionable tips for developers. Our goal is to provide insights and practical guidance for those looking to leverage smart contracts in secure, scalable, and efficient digital asset custody solutions[1].

By Vince Liu, Lead Engineer and Mohammad Nauman, Director of Engineering

At Bullish, innovation and technological advancement are at the core of our platform. From day one, we have embraced blockchain technology to enhance the robustness of our ecosystem, and a key pillar of this journey has been the strategic implementation of smart contracts. These self-executing agreements, embedded with predefined rules and driven by deterministic execution, have redefined how we approach automation, security, and operational efficiency. Deployed on our private **Taurus blockchain**, the smart contract layer has become a cornerstone of Bullish's approach to building a more secure, reliable infrastructure, enabling us to redefine the standards for blockchain-based automation by ensuring that each process runs predictably, consistently, and more securely.

This blog delves into Bullish's hands-on experience with smart contracts, offering an in-depth look at the challenges we've encountered, the innovations that have propelled us forward, and the key lessons we've learned along the way. By sharing our journey, we aim to provide practical insights to fellow developers and engineers, shedding light

on how smart contracts can unlock powerful opportunities in secure, automated environments.

1. How smart contracts drive Bullish business processes

Building on this foundation, Bullish leverages smart contracts to transform key business processes, delivering improved security, automation, and reliability at scale. These self-executing agreements ensure cryptographically verifiable transactions, creating a more efficient and secure environment for our operations. Here's how smart contracts are helping Bullish enhance various critical processes:

1.1 Strengthening security through automation

Smart contracts provide a more secure backbone for numerous vital components, including custody services, authentication protocols, and operational workflows. By ensuring that each transaction and process is validated within a cryptographically secure environment, smart contracts minimize the risk of tampering or unauthorized access. This automation of security controls helps Bullish maintain high system integrity without the need for manual checks, further reducing human error.

1.2 Reducing manual errors with precise automation

One of the key advantages of smart contracts is their ability to automate complex workflows with precision. Deterministic execution ensures that predefined conditions trigger processes automatically, significantly lowering the need for human intervention. This not only accelerates processes but also reduces the likelihood of data entry errors or delays caused by manual tasks. The result is a smoother, more reliable operational flow.

1.3 Establishing a single source of truth

In systems where data accuracy and provenance are critical, smart contracts act as an immutable, cryptographically signed repository of information. On public blockchains, immutability generally means that once a transaction is written, it cannot be altered. In contrast, on private blockchains like Taurus, immutability refers to the integrity and cryptographic verifiability of records. While updates can occur under strict governance, the original transaction history remains intact and traceable. By acting as a single source of truth, smart contracts ensure sensitive data remains tamper-proof and accessible only to authorized parties, maintaining both compliance and data integrity across Bullish's processes.

1.4 Boosting transparency with built-in auditability

Smart contracts inherently provide auditability due to the blockchain's time-stamped and immutable nature. Every

transaction and data update is recorded, making it possible to trace back through a clear audit trail. This transparency increases accountability across the board, offering an unparalleled framework for ensuring compliance and accuracy in every process.

2. A typical use case: Automating transaction verification with smart contracts

A core application of smart contracts within Bullish is the automation of transaction verification, ensuring that each transaction is processed in a more secure, transparent, and efficient manner. By leveraging a combination of smart contracts, oracles, and external AML/KYT providers, Bullish can ensure that transactions are not only cryptographically verified but also comply with regulatory requirements.

The following steps outline a typical workflow for processing transactions:

A new transaction is recorded by the smart contract.
The process begins with a new transaction being initiated and logged in the smart contract. At this stage, the transaction details are cryptographically signed and immutable within the system, ensuring that any subsequent actions are traceable and secure.
The oracle detects the new transaction.
Once the transaction is recorded, the oracle—a bridge between on-chain and off-chain data—detects the new transaction entry from the smart contract's data table. The oracle plays a key role in ensuring that external data sources can be integrated seamlessly with the blockchain.
The oracle sends the transaction for off-chain checks.
After detecting the new transaction, the oracle sends the transaction details to an external AML/KYT provider. This step ensures that the transaction undergoes necessary anti-money laundering (AML) and know your transaction (KYT) compliance checks, adding an additional layer of security and regulation.
The provider returns an updated score.
The AML/KYT provider assesses the transaction, generating a risk score based on the transaction's legitimacy and compliance with regulatory standards. This updated score is then sent back to the oracle.
The oracle updates the transaction score in the smart contract.
Upon receiving the updated score, the oracle pushes this data back to the smart contract, where the transaction's status is updated accordingly. The smart contract ensures that only authorized services, like the oracle, can modify transaction details, maintaining the integrity of the entire process.
The transaction proceeds to the next step or check.
Based on the updated transaction score, the smart contract determines whether the transaction should proceed to the next stage of the workflow or be flagged for further review. This loop continues until the transaction passes all necessary checks and is fully processed.

This use case demonstrates the power of smart contracts to integrate multiple data sources and automate complex workflows with cryptographic security, ensuring both precision and compliance. By combining blockchain's inherent security with external verification systems, Bullish creates an environment in which every transaction can be handled with precision and compliance.

However, an important question remains: How can we trust the oracle in this process? While oracles act as the crucial bridge between on-chain and off-chain data, ensuring the security and reliability of this interaction is key to maintaining trust in the system. In future blogs, we'll explore this topic in depth, focusing on how Bullish addresses the challenges of oracle trust and integrity within our broader infrastructure.

3. Overcoming key challenges in smart contract implementation

Implementing smart contracts at Bullish was a transformative journey, one that tested our ability to adapt to evolving business requirements, maintain compliance with stringent regulatory standards, and integrate these contracts into a complex distributed system. Each challenge provided an opportunity for growth, pushing our team to develop creative, technical solutions that have ultimately enhanced the robustness of our platform.

3.1 Adapting to evolving business requirements

In the fast-paced world of digital assets, business needs are constantly changing. While public blockchains often treat smart contracts as immutable, this rigidity is less practical in a business environment where requirements frequently evolve. Bullish faced this challenge head-on by adopting an iterative development process for our smart contracts, allowing us to remain agile while maintaining operational efficiency and security.

Our approach involved designing modular smart contracts that could be updated or extended without disrupting core operations. By breaking down contracts into smaller, self-contained modules, we allowed for flexibility in specific functions while ensuring that the integrity of the larger system remained intact. This enabled us to adapt our contracts as business requirements shifted—whether through new asset types, evolving market conditions, or regulatory changes—without needing to start from scratch.

3.2 Enabling compliance and meeting regulatory standards

Handling sensitive data is one of the biggest challenges in the digital asset space, and at Bullish, compliance with privacy and regulatory standards is paramount. One key difficulty we encountered was ensuring that our smart contracts adhered to strict data governance rules, particularly the requirement that no Personally Identifiable Information (PII) be stored directly within the blockchain.

To address this, we worked closely with our compliance and data governance teams to architect smart contracts that balanced the need for transparency and accountability with stringent privacy protections. By segregating sensitive data from on-chain processes and using off-chain storage for PII, we enabled our contracts to remain compliant while still maintaining the integrity of the underlying data.

This collaboration extended beyond engineering, involving multiple stakeholders from various departments. It required constant communication and refinement to ensure that our contracts adhered to regulatory frameworks while continuing to meet operational needs.

3.3 Navigating system integration complexities

Integrating our smart contracts with Bullish's distributed systems presented technical challenges due to the use of a protocol not part of common frameworks. This added complexity to the process of ensuring seamless operation within our infrastructure. To address this, we built robust APIs that facilitated communication between smart contracts and both on-chain and off-chain systems.

We also prioritized backward compatibility so that future versions of our contracts could still interact with legacy systems, minimizing disruptions during upgrades. Rigorous testing protocols, simulating real-world scenarios, allowed us to maintain system integrity while deploying new iterations.

By implementing a controlled deployment pipeline with rollback capabilities, we minimized downtime and enabled system stability. Continuous monitoring helped us catch and address any issues post-deployment, maintaining the security and reliability of our system.

4. Key engineering insights from smart contract implementation

Throughout our journey with smart contracts at Bullish, we encountered several valuable lessons that have shaped our approach to development and implementation. These insights reflect both technical challenges and strategic decisions, helping us refine our methods and improve the robustness of our platform.

4.1 Data segregation is crucial

- Challenge: Managing sensitive data while minimizing security risks is a top priority, especially when dealing with distributed systems. Ensuring that different types of data are isolated and handled appropriately was a critical consideration in our smart contract design.
- Solution: We adopted a data segregation approach, using modular smart contracts to create distinct silos for different data types. Each contract handled only the data necessary for its specific function, reducing the

potential blast radius in case of security incidents. By segregating sensitive data, we minimized exposure and improved the manageability of the system.

- Impact: This approach not only increased the overall security of our platform but also improved system efficiency by reducing the complexity of data management. Each contract's focused responsibility allowed us to streamline operations and enhance protection for sensitive information.

4.2 Friendly and backward-compatible interfaces: A major win

- Challenge: As Bullish evolved, it became crucial that new smart contracts could seamlessly integrate with legacy systems without introducing incompatibilities or operational bottlenecks.
- Solution: We designed interfaces that prioritized backward compatibility and developer-friendly interactions. These interfaces allowed smart contracts to evolve without breaking existing functionality. We also applied extensive testing to verify that newer contracts worked flawlessly with older systems, which minimized disruptions during upgrades.
- Impact: By focusing on backward compatibility, we achieved smoother integrations and reduced the operational overhead associated with deploying new versions of smart contracts. This flexibility has been critical in maintaining the efficiency of our platform as it grows.

4.3 Striking the right balance with smart contract application

- Challenge: A key lesson we learned was understanding when not to use smart contracts. While they offer significant advantages, overusing them in areas where simpler solutions suffice could introduce unnecessary complexity.
- Solution: We adopted the principle of "Less is More" when applying smart contracts and blockchain solutions. This principle guided us to strategically implement smart contracts only in areas where they provided clear value—such as automation, security, and auditability—while avoiding them in areas where traditional methods were more efficient.
- Impact: By embracing this selective approach, we reduced operational complexity and optimized performance. Smart contracts were applied only where their benefits were most impactful, resulting in a leaner, more efficient system.

4.4 Navigating internal data evolution in smart contracts

- Challenge: As business requirements evolved, managing changes to the data schema within our smart contracts without causing disruptions was a complex technical challenge.
- Solution: We implemented schema versioning and carefully planned migrations so that changes to the smart contract data structure were backward-compatible. By employing a phased approach to upgrades, we were able to maintain data integrity and minimize risks during transitions.

- Impact: This careful handling of data evolution allowed us to smoothly upgrade contracts while preserving historical data and ensuring continuous operation. It also reduced the risk of disruptions during schema changes, which helped maintain the reliability of our platform.

These key engineering insights reflect our commitment to continuously refining our approach to smart contract implementation. Each challenge presented an opportunity to learn and innovate, helping Bullish build a more secure, scalable, and efficient platform for the future.

5. Technical innovations driving smart contract evolution

Bullish's journey with smart contracts has been marked by key technical innovations that have significantly advanced the way we develop and operate within our blockchain ecosystem. Each breakthrough addressed critical needs, simplifying complex processes and improving overall system efficiency.

5.1 Replacing ABIs with Protobuf for efficiency

Bullish transitioned from traditional ABI interfaces to Protobuf to overcome the complexity associated with consuming smart contracts. ABIs, while functional, required special serializers and introduced unnecessary friction in the development process. Protobuf, with its built-in versioning support and backward compatibility, provided a cleaner, more efficient way to interact with smart contracts. By decoupling the contracts from their consumers, we simplified the process of contract upgrades and maintenance. The result was a more agile development pipeline, allowing us to iterate quickly and implement changes without disrupting existing functionality.

5.2 Smart contract SQL-like query interface: Making data more accessible

One of the major challenges in working with blockchain data is the difficulty of accessing smart contract state in an intuitive manner. To address this, Bullish developed an SQL-like query interface that allowed engineers and operators to interact with the blockchain state using familiar SQL syntax. This innovation significantly lowered the barrier for non-blockchain developers, enabling them to work with blockchain data as if it were a relational database. The ease of querying and accessing data in this way made cross-functional collaboration smoother and more efficient, bridging the gap between blockchain and non-blockchain systems.

5.3 Synchronized executions: Enabling real-time updates

Traditional blockchain systems often suffer from sequential execution models that prevent actions from being executed within other actions, similar to function calls. Bullish addressed this limitation by implementing synchronized executions, allowing nested actions to occur seamlessly during smart contract execution. This innovation was particularly important for workflows requiring immediate consistency, ensuring that the system

could handle complex interactions during function execution. By enabling synchronized executions, we reduced data conflicts and improved the responsiveness of our system, delivering faster transaction processing and more accurate outcomes.

5.4 Linter for EOSIO protocols: Reducing human error

Given the complexity of working with EOSIO protocols, human error was a constant risk during the development of smart contracts. To mitigate this, Bullish created a custom linter specifically tailored to EOSIO, which helped catch common errors early in the development process. This tool became an essential part of our development workflow, reducing debugging time and improving the overall quality of our smart contracts. The linter allowed our engineers to focus on higher-level development tasks while ensuring that the underlying code adhered to EOSIO standards, resulting in more robust and error-resistant contracts.

5.5 Enhanced error handling and logging: Intuitive codes and structured logs

Efficient error handling and monitoring are critical for maintaining the reliability of smart contracts. Bullish enhanced its error handling mechanisms by introducing more intuitive error codes and moving to structured logs. These changes allowed engineers to quickly diagnose and resolve issues, leading to faster incident response times. Structured logs also provided better insights into smart contract performance, enabling more sophisticated analysis and monitoring. By improving our logging and error handling processes, we enhanced the operational oversight of our smart contracts, leading to a more reliable and stable system.

These innovations have been instrumental in advancing Bullish's smart contract capabilities. From simplifying interfaces with Protobuf to enabling real-time execution and improving developer workflows with tailored tools, each breakthrough has contributed to a more efficient, scalable, and secure platform. As we continue to push the boundaries of blockchain technology, these innovations will serve as the foundation for future developments in our ecosystem.

6. Actionable tips for enterprise smart contract developers

6.1 Plan for scalability from the outset

Scalability is a critical factor in smart contract design. Whether your platform will handle hundreds or millions of transactions, planning for scalability ensures that your contracts don't become bottlenecks as the system grows. At Bullish, we implemented modular smart contracts that could scale as needed by handling specific tasks without overwhelming the broader system. Designing contracts for scalability from the start reduces the need for major rewrites later.

6.2 Maintain version compatibility

Preserving backward compatibility is essential when updating smart contracts, especially in environments where older versions still interact with newer ones. Bullish adopted careful versioning strategies so that new smart contracts could interact seamlessly with legacy systems, minimizing disruptions. This approach also helped in scaling up without overhauling the entire infrastructure, allowing for smoother upgrades over time.

6.3 Regular data pruning to maintain efficiency

As smart contracts execute, residual data can accumulate and potentially slow down performance. Preemptive data pruning helps maintain contract efficiency and keeps storage costs low. By proactively identifying and removing unnecessary data, Bullish was able to optimize smart contract performance over time and ensure that our system remained responsive.

6.4 Comprehensive testing as a foundation

Smart contracts are often immutable once deployed, making testing a critical part of the development process. At Bullish, we employed a combination of unit tests, integration tests, and stress tests to ensure our smart contracts could handle real-world scenarios before going live. Investing in robust testing ensures that any issues are caught early, reducing risks in production.

6.5 Build thorough release processes

The process of releasing smart contracts into a production environment requires careful planning and attention to detail. Bullish established a well-defined release pipeline with detailed review and approval processes. This helped ensure that contracts were thoroughly vetted, reducing the risk of post-deployment issues.

6.6 Invest in efficient tooling

The right tools can greatly enhance the speed and efficiency of smart contract development. Bullish invested in custom tooling, such as our EOSIO linter and structured logging systems, to improve code quality and error handling. Having the right tools in place streamlines development and reduces time spent debugging or tracking down issues after deployment.

6.7 Security should always be a priority

Security is a top priority for any blockchain operation. Bullish worked closely with security teams with the goal of having all smart contracts adhere to the highest standards of cryptographic security. Collaboration between developers, auditors, and security experts aimed to make contracts as secure as possible prior to deployment.

Conclusion

Bullish's journey in advancing smart contract technology has been both challenging and rewarding. Throughout this

process, we've addressed evolving business needs, navigated compliance with regulatory standards, and innovated across various technical fronts. From designing modular contracts to enhance scalability, to integrating robust interfaces and simplifying complex workflows, our smart contract implementation has been built on a foundation of security and operational excellence.

As we look ahead, Bullish will continue to push the boundaries of digital asset custody technology. The innovations we've implemented, such as Protobuf-based interfaces, SQL-like queries, and synchronized executions, represent just the beginning. Our commitment to maintaining secure, scalable, and high-performing systems will guide our next steps, as we continue to refine and evolve our smart contract framework. Each lesson learned and every challenge overcome has strengthened our platform and positioned us to lead the way in future blockchain advancements.

Feeling inspired by the cutting-edge work we're doing at Bullish? We're always on the lookout for talented individuals to join our team. **Explore our open roles** and be Bullish on your career. Want to stay up to date with the latest news from across Bullish? Follow us on **LinkedIn** and **X**.

[1]Custody solutions provided by members of the Bullish Group (1) may differ in particular jurisdictions to meet regulatory and customer requirements and expectations; and (2) will continue to evolve over time, so the information in this blogpost may become outdated. Please reach out to support@bullish.com if you would like to understand the approach currently used to protect customer assets in your jurisdiction.

ARE YOU BULLISH?

Take charge and be Bullish on your career by helping us build the best digital asset trading platform for institutions.

Work at Bullish