FidesInnova
A platform for decentralized trusted IoT systems

FIDESINNOVA SERVICE CONTRACT

Service Contract in zk-IoT: Automating Data Sharing

FIDESINNOVA • October 3, 2024 • 0 Comments

Service contracts automate and manage agreements between different IoT devices and users in the FidesInnova ecosystem, enabling them to share data or function based on other IoT devices' data. These contracts allow IoT devices to expand their functionalities and actions based on data received from other devices, enable users to trust data from IoT devices owned by others, and automate the data-sharing process.

1. How FidesInnova's Service Contracts Enable Seamless Data Sharing?

Service contracts are script-based codes designed to encode business logic between IoT devices, allowing them to communicate and share data without the need for a central authority. Similar to smart contracts in blockchain technology, these contracts operate specifically within the decentralized IoT space. They are automated, self-executing, and enforceable through code, ensuring secure and transparent transactions and interactions between parties, all without intermediaries.

Executed on FidesInnova IoT nodes, service contracts utilize a virtual machine (VM) capable of running JavaScript. This enables the creation and execution of complex interactions between IoT devices and users, enhancing adaptability and expanding functionality within IoT ecosystems.

2. Key Features and Components of Service Contracts

- **Visual and Text-Based Development:** The FidesInnova Web App accommodates both novice and experienced developers, allowing service contracts to be created using either Blockly or JavaScript. The Visual Console, utilizing Blockly, offers a graphical development method, while the Text-based Console, based on JavaScript syntax, caters to traditional coding practices.
- **Zero-Knowledge Proof (ZKP) Support:** Any data received from IoT devices coded in a service contract can contain a Zero-Knowledge Proof (ZKP) that demonstrates the accuracy of the code execution on the IoT device. The service contract includes an internal ZKP verifier that automatically verifies the received ZKP before executing the logic based on the data or sending a command to another IoT device.
- **MQTT Protocol Support:** The development environment can receive data and send commands to any IoT device that supports the MQTT protocol.
- **Blockchain Integration:** Service contracts are integrated with blockchain technology to leverage decentralized and secure networks. This ensures that all agreements coded in a service contract, along with devices and ZKPs, are transparent, immutable, and tamper-resistant. The blockchain also verifies data collected from zk-IoT devices, enhancing system reliability.
- **Data Management and Monetization:** Users can share or sell data collected from IoT devices (e.g., temperature, humidity, air quality, and noise levels) through automated service contracts that define how data is collected, accessed, and monetized.

- **Decentralized Service Contract Execution:** FidesInnova leverages ZKP technology to enable any IoT server in its network to execute a service contract. This decentralized execution eliminates the need for a trusted IoT server, improving efficiency and reducing costs by minimizing intermediaries. To ensure optimal performance, FidesInnova integrates the isolated-vm technology. This isolation prevents errors or resource leaks in service contracts from affecting the overall system. Additionally, FidesInnova creates multiple isolated contexts within a single Node.js process to prevent runaway code from consuming excessive resources, allowing independent execution of multiple service contract instances and enhancing system stability.

3. Functionality and Use Cases

- **Automated Building Management:** Service contracts can be customized to manage various building environment sensors, such as air quality control, noise monitoring, temperature, and humidity. These contracts automate processes by linking real-time sensor data to specific actions, streamlining management and reducing the need for manual intervention. For example, a service contract could be programmed to send a notification to the strata company if noise from a unit or apartment persists for more than 30 minutes after 9:00 PM.

- **IoT Device Communication:** Service contracts enable IoT devices to communicate autonomously. For example, if a building's multi-sensor detects unsafe CO2 levels, the service contract can trigger an alert or automatically adjust ventilation systems, enhancing safety and reducing response times. Similarly, if the house temperature exceeds 40 degrees Celsius, a service contract can trigger a siren on a speaker. Additionally, if high temperature and humidity persist for an extended period, posing a mould risk to the building, the system can send a notification to the owner or management company.

- **Data Visualization:** The TrustSense project, built on FidesInnova technology, is an example of environmental data visualization. It allows users to share their devices and create a visual map of their environmental data. Currently running in British Columbia, Canada, the project can be accessed at www.trustsense.tech

- **Gas Detection and Response System:** The EnergyWiseNetwork project, built on FidesInnova technology, exemplifies gas sensor automation. Service contracts are crucial in this context for automating and managing the monitoring of critical parameters such as gas leak detection, environmental safety, and equipment performance in real time. The project can be accessed at **www.energywisenetwork.com**

4. Service Contract Example: Noise Monitoring and Management in Residential Buildings

Objective: To deploy a service contract that monitors noise levels in a residential building. The service contract will trigger actions such as notifying building management when noise exceeds a predefined threshold and taking corrective measures, such as issuing warnings to tenants.

Steps for Deploying a Service Contract on the Blockly Platform:

1. Define the Conditions (Noise Level Monitoring):
The service contract is programmed to monitor noise levels using multi-sensors installed in common areas and individual units of the building.
A noise threshold (e.g., 70 decibels, commonly associated with loud conversations or party noise) is set as the condition for triggering the contract.

2. Set Data Inputs (Real-Time Noise Data):
The sensors send real-time noise level data to the TrustSense platform, which aggregates and validates the data to ensure only accurate and secure data is used to trigger the service contract.
Additional data, such as the time of day and the location within the building where the noise is detected, is also captured for context.

3. Define the Responses (Automated Actions):

- **Primary Response:** If noise levels exceed the threshold, the service contract automatically sends a notification to the building manager via the TrustSense app or web app dashboard.
- **Secondary Response:** If the high noise persists or exceeds a critical level (e.g., 90 decibels), the contract triggers additional actions, such as sending an automated warning message to the responsible tenant or adjusting soundproofing systems in the affected areas.
- **Data Logging:** The service contract logs the noise event on the blockchain, creating an immutable record for future reference or disputes. This can be used to address tenant complaints or monitor chronic noise issues.

5. Deploy the Service Contract:

Using the Blockly platform, the service contract is built by dragging and dropping visual blocks representing the conditions and actions defined above. Once the contract logic is completed, it is deployed on the blockchain. The deployed contract now autonomously monitors noise levels and responds to real-time data from the sensors.

6. Steps to Deploy Your Desired Service Contract:

1. Go to the FidesInnova website.
2. From the Product menu, select the Web App option.
   In the Web App menu, choose the specific web app you need from the FidesInnova ecosystem and access it.

3. In the dashboard, select the Create New Service option.

4. In this section, input the details of your Service Contract and click the Create button at the end.

5. Then, in the dashboard, navigate to the My Services section to view the created service contract.

6. From the Code menu, enter the Blockly environment, where you can program and create the code related to your service contract.
7. Once you have finished, click Save to create your service contract.

Service Contract Sharing Data zk-IoT

‹ PREVIOUS

Decentralized Delegated Proof-of-Stake (D2PoS)

NEXT ›

Service Market

Leave a comment

Your Name *

Your E-mail *

8. Finally, by pressing the Publish button from My Services men, your service contract will be submitted for final approval by the admin and then published to the Service Market.

☐ Save my name, email, and website in this browser for the next time I comment.

Your comment *

☐ I agree that my submitted data is being collected and stored. For further details on handling user data, see our Privacy Policy.

I'm not a robot

reCAPTCHA
Privacy - Terms

Leave a comment

You May Also Like

FIDESINNOVA

Unlocking the Power of Digital Twins in Smart Building Management

FIDESINNOVA, SERVICE CONTRACT

Service Market