Nodle Network.
Smart Missions Paper

Connecting Web3 to the Physical World: Nodle's Smart Missions

A new generation of smart contracts
for the mobile computing platform

nodle

Foreword

The past years spent developing Nodle were essential to assemble the building blocks of the ecosystem. Reaching many significant milestones:

December 2018

Releasing the first Proof of Concept of the network on the Stellar Network, with native support on the HTC Exodus 1 and 1s, the blockchain phones from HTC..

June 2020

Nodle was among the first teams to launch a full-production blockchain built with Parity Substrate.

May 2022

The project migrated from its solo chain to the 11th parachain slot on the Polkadot Network, leading the way of innovation before many others would take a similar path.

At the same time, the network has been constantly optimized and hardened against potential threats. In addition, the rewards formula and system have been improved to incentivize network growth and become more transparent, thus providing better economic guarantees to NODL token holders.

The Nodle App, which has been a critical component since day one, showcases the network's capabilities, allowing users to experience the power of Nodle. In 2018 it was one of few Stellar-compatible wallets, and when the Nodle chain launched in 2020, it also became the third mobile wallet to support Parity Substrate / Polkadot.

The team behind Nodle are outliers, forethinkers, and innovators. Even if developments started five years ago, the work is just beginning. A fascinating fact is that as the network expands, it becomes increasingly valuable to builders worldwide. This is why new capabilities are being introduced today as Smart Missions (also referred to as “Missions” below).

Smart Missions are the core primitive of the Nodle Network. It allows anyone to program the network's swarm of smartphones and devices to execute a custom piece of code or action for a given context. It makes the network programmable, meaning that developers and builders can come in and easily create applications on top of the Nodle infrastructure in exchange for a fee in NODL tokens. New features and products will be built through Smart Missions such as asset tracking, data collection, security checks, geographic NFT airdrops, social networking, or referrals.

The Nodle team is already building additional capabilities and Smart Missions use cases, showcasing the network's potential. Third-party builders will create more services on top of the network for their ideas and products.

Table of contents



Smart Missions are the “primitive” of the Nodle Network.

Other applications, use cases, or features are built as Smart Missions.

This is possible because of the combination of various pieces of technology:

The Nodle SDK. Deployed within mobile applications across hundreds of thousands of devices, it allows the Nodle Network to communicate with the real world by using smartphones running the SDK[1].

The Nodle Parachain and Parity Substrate. Acts as the nervous system that links the multiple components of the Nodle Network together and handles settlements (i.e., payments) for usage on the network.

The Nodle Virtual Machine (VM). A common and standardized system to represent and run scripts across various mobile platforms with reasonable performance and maximum security.

[1] For privacy and security reasons, SDK-enabled nodes should be able to filter out what commands they accept from the network. This can be done at the SDK integrator level, or at the level of the Node Operator, which operates fleets of SDKs.



Bird's eye view

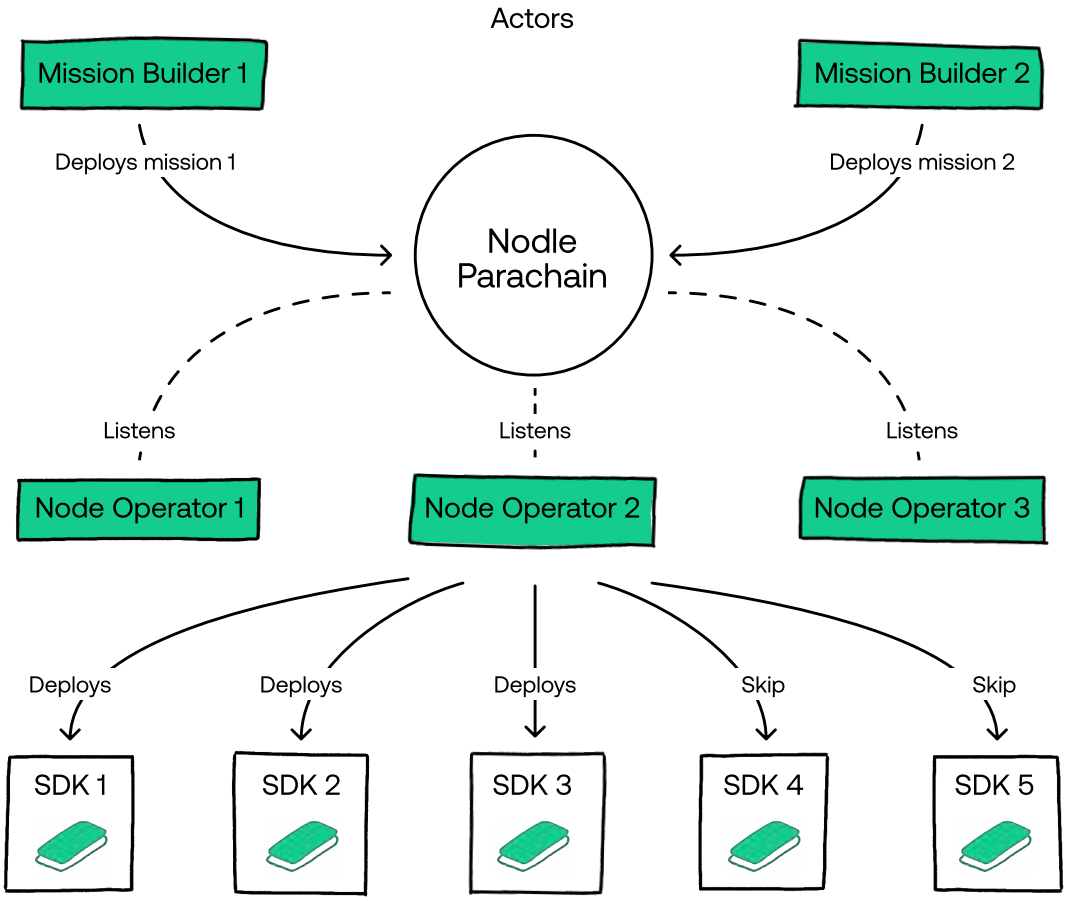
A Smart Mission is composed of two main components:

- The first one runs on devices at the edge with the Nodle SDK that embeds an implementation of the Nodle VM, let's call it the device code.
- The second one runs on the Nodle Parachain in the form of a Wasm smart contract; let's call it the chain code.

The Nodle Parachain maintains a registry of Smart Missions that are deployed by builders that want to use the network's resources. The chain code portion of a Smart Mission exposes standardized APIs (the "Standard Mission Interface"). This is similar to Ethereum's ERC standards which create standardized calls and APIs for special smart contracts like tokens. These standardized calls are used to identify and classify each Smart Mission before they are deployed to Nodle SDK instances.

Nodle SDKs[2] do not query the parachain directly. Instead, they rely on another server, a Node Operator, which manages a fleet of SDKs. The Node Operator knows which SDKs are online and select which Smart Missions should be deployed to which SDK instances. It might also enforce its own security, economics, or compliance requirements on such missions and their builders.

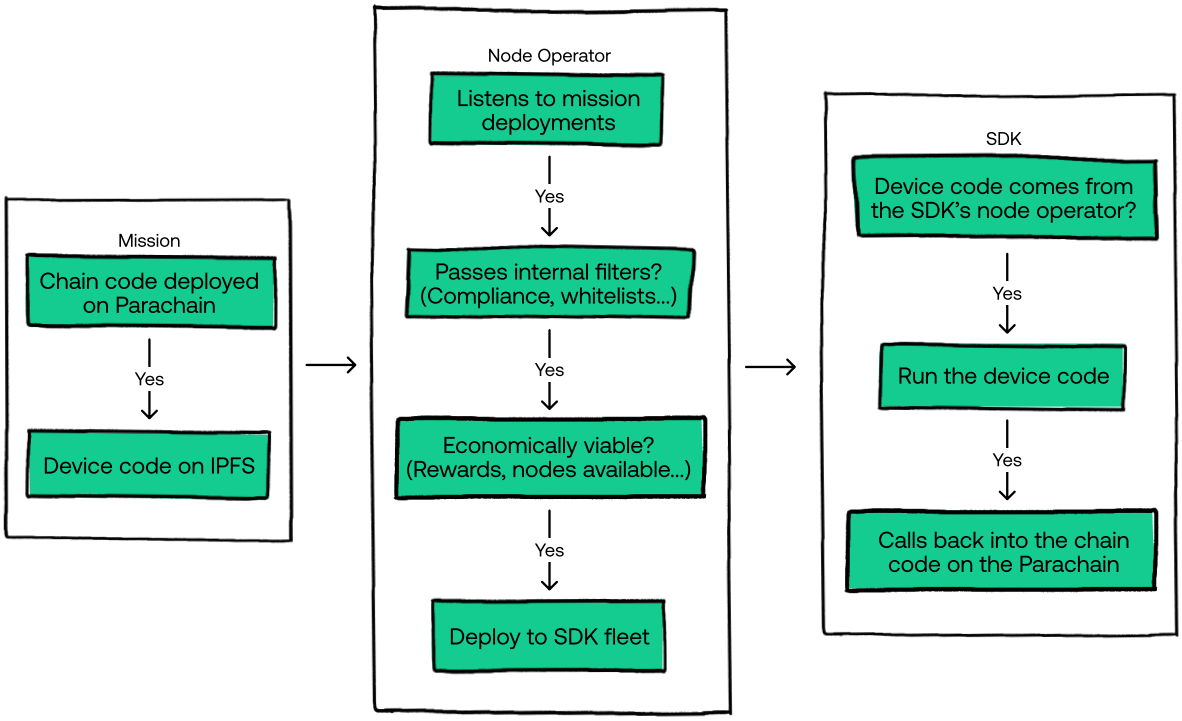
[2] In the case of the Nodle App and similar solutions, the user is in control and might choose by themselves missions they want to run.



The device code is downloaded by Node Operators and deployed to their fleet of Nodle SDKs, which may execute it. When executed, the device code has access to a variety of APIs exposed by the Nodle VM environment (the "MobileVM API") and can run the necessary computing defined by the mission builder. It also has the ability to call back into the chain code with an arbitrary set of parameters. These parameters will very often be a "proof": a piece of data that the chain code can verify, or an "attestation": evidence of a certain fact; or may simply rely on a centralized oracle. The chain code will verify this data and unlock some assets as compensation for the Node Operator or Nodle App user.

The Nodle Network will expect device codes to be stored on the InterPlanetary File System[3] (IPFS). IPFS has proven itself as a reliable and decentralized file storage network. It is well-adopted within the Web3 industry to host pictures of collectible cats or monkeys for non-fungible tokens, websites, and more. In fact, it is already used across the Nodle Parachain to support

[3] https://ipfs.tech/



mobile-generated NFTs. IPFS associates each document on the network with a unique identifier in the form of a cryptographic hash. This allows the chain code to indicate which device code needs to be run to participate in the mission. Because the device code is identified by its hash, this also allows Node Operators to verify that they are manipulating the correct and untampered device code.

Initial implementations of the Nodle VM support a custom scripting language optimized to be safe, memory efficient, and simple to use. The chain code will be run via Parity Substrate's contracts[4] pallet or a variant, and support the ink![5] domain-specific language (based on the rust language with some extra syntactic sugar), or any other language that compiles to Wasm with the correct APIs exposed, such as ask![6] (which uses AssemblyScript[7], a TypeScript-like language for Wasm).

For security and transparency purposes, a Smart Mission needs to specify a set of permissions required to run its device code. When a mission's device code is run, the Nodle VM is in charge of enforcing these permissions. This can also be used by Node Operators to filter out which Smart Missions they are willing to support or to better represent a mission's needs and accesses to a Nodle App user.

[4] https://github.com/paritytech/substrate/tree/master/frame/contracts
[5] https://ink.substrate.io/
[6] https://ask-lang.github.io/ask-docs/
[7] https://www.assemblyscript.org

Difference with Existing Systems

One will notice some similarities between Smart Missions and other systems, such as existing Decentralized Applications (DAPPs) platforms like Ethereum and Solana, or various distributed computing platforms.

It is key to understand that these other platforms were designed to either support verifiable (Smart Contracts) or high-performance computing applications. Nodle is different, Smart Missions are a way to develop Web3 applications that interact with the real world. Meaning that one can write a piece of code designed to be executed on a remote device such as a smartphone and which may interact with surrounding Bluetooth devices or perform an arbitrary task with access to the device's sensors such as GPS, camera, or accelerometer.

In that sense, Nodle isn't a DAPP platform, but rather a bridge between the real and digital worlds.

Builder Ecosystem

When creating a platform where builders and developers can deploy and run code, builder experience is key. Builders will need strong support for the tools they already use such as their code editor, debugger, or linter.

Builders will want to test their Smart Missions in a production-like environment and on physical Nodle-enabled devices. This is where the Nodle App comes in. The Nodle App is a special way to run the Nodle SDK while offering its users more control. The Nodle App can download and run Smart Missions selected by its user, whereas most other apps with the Nodle SDK will delegate this responsibility to a third party. Indeed, Node Operators are accountable for enforcing privacy regulations such as the CCPA and the GDPR; they will select which Smart Missions can run on the nodes under their responsibility. The Nodle App will then play a critical role as a development tool for builders and as the primary way to interact with Smart Missions targeting end-users.

The builder should be able to use the Nodle App to scan a QR Code on their computer, instructing it to download the Smart Mission's device code and seamlessly run it for testing purposes. This simple development will make it much easier for builders to test their missions in production-like conditions or share them with people. Over time, the Nodle App will provide statistics, additional options, and APIs to help debug the Smart Mission.

Mobile VM API

It was previously mentioned that the Nodle VM gave access to various APIs on the hosting device. Indeed, a mission needs to interact with the outside world by accessing data from the Bluetooth chipset or local details like the user's wallet address. It also needs a way to call back into the chain code to trigger its compensation. These APIs can be exposed directly inside the VM.

The following calls will be supported by the Nodle VM:

Bluetooth Low Energy APIs: broadcasting of beacons in various formats, scanning of nearby devices, GATT queries...

Wallet and Parachain addresses: will offer separate calls to query the host's address, the SDK integrator address, or the mission chain code's address

Attestation: will rely on the SDK's built-in protection mechanisms and its interactions with the Nodle Node Operator to return an attestation that the SDK is running on a device considered genuine.

Callback: given an arbitrary set of parameters and a chain code address, this will trigger a call back to the Nodle Parachain.

Internet access: if the device code was granted the appropriate permissions, it should be able to communicate with a remote server hosted somewhere else on the Internet. This could take the form of a normal HTTP request or more complex networking schemes.

Chain Runtime Extensions

The chain code is the component the device code calls back into. The chain code takes the form of a Wasm Smart Contract running on the Nodle Parachain. As such, it can access various on-chain primitives which will allow mission builders to create new use cases, economics, and reward mechanisms:

Native NODL transfers: allows checking one's balance, transferring tokens held in the chain code contract, requesting payments in NODL, etc.

Third-party assets: allows creating and managing a new token, transferring tokens held in the chain code contract, requesting payments with third-party assets, etc.

Non Fungible Tokens (NFTs): used to issue or request NFTs present on the Nodle Parachain.

Mission Registry: allows querying the registry for currently deployed missions and their metadata or deploying a new Smart Mission.

Interoperability primitives (XCM): supports calling functions on other parachains and sending tokens or other assets across the Polkadot network.

Standard Mission Interface

Each Smart Mission's chain code must expose a set of functions and storage values that Node Operators will use to filter out which Smart Mission they will execute[8], and by the Nodle App to correctly classify each Smart Mission. While this interface will further be defined as the network continues being developed some basic requirements can already be highlighted:

- A function that returns the IPFS document hash identifying the associated device code. This allows Node Operators and Nodle App instances to download the appropriate code to perform the Smart Mission. As covered previously in the Bird's Eye View, every device code needs to be made available through IPFS.
- A function that returns the IPFS document hash of a metadata file. Such metadata shall include:
  - A short name
  - A longer form description
  - Some logos to be displayed on various frontends
  - Links to the developer's website for more information
  - The type and amount of rewards one collects, such as a certain amount of tokens, or special NFTs
  - The Smart Mission's high-level category such as whether it is an asset tracking mission or a positional mission which is restricted to a certain area

[8] It is assumed that each Node Operator might also have its own logic to choose which mission to deploy and where. For instance, they might perform static analysis, or require special whitelisting logics for certain usage scenarios.

- Depending on the Smart Mission's category, the metadata should contain additional parameters:

  For asset tracking, it should contain a target filter which filters what kind of devices the developer is looking for as well as an optional geographic bound.

  For positional missions, the developer must specify the geographic bound in which the mission is available.

- Nodle VM requirements, such as the memory necessary for the execution of the device code.

- Permission requests that clearly highlight the access offered to the device code during its execution. When the device code is run, the Nodle SDK should configure the Nodle VM to only allow the requested permissions.

Depending on the metadata registered or the requirements of the device code, Node Operators might use their own algorithms to decide whether a Smart Mission should be supported by them and if so, do it in the most optimal manner. For instance, a geographically bounded mission shouldn't be sent to a smartphone that is too far away from the target area but could still be downloaded ahead of time as the user gets closer to the target area.

It is expected that the Nodle App will obey the Nodle Node Operator operated by the team, especially for asset tracking missions as they can be executed passively. However, it should also let the user specify its filters or select missions manually[9], which would be quite convenient for Smart Missions that require an action from the end user. This could be the case for a mission that requires the user to go to a specific place or perform a special action.

[9] Going forward, it will be assumed that this statement is implicit whenever Node Operators or Nodle SDKs are discussed.

Nodle VM Inner Workings

The Nodle VM is a key component of the Smart Missions system; it is in charge of exposing the Mobile VM API to the device code in a sandboxed environment. At first, it will be backed by a simplified scripting language built with ANTLR[10]. This language will be generic enough to support the first missions running on the Nodle Network by expressing conditions, filters, and calls to sub-modules of the Nodle SDK.

As the ecosystem's needs arise and evolve, it will be necessary to extend the Nodle VM with more general capabilities and possibilities for builders. As such, it is intended to embed a Web Assembly (Wasm)[11] interpreter within the Nodle SDK. Because Wasm is a well-defined, battle-tested, binary instruction format and VM, it will mean that builders can easily develop missions using the tooling of their choice. The usage of a Wasm-based VM will allow the Smart Missions to natively support standard languages used by millions of developers across the globe: JavaScript, Golang, Rust, C, and more... Because of this, it will reduce the need for customized cooling such as testers, linters, and debuggers; thus allowing the team behind Nodle to focus on their core innovation: bridging the gap between the physical and digital worlds.

It was chosen to start with a simplified scripting language to better start the inherent Nodle VM and put it in the hands of the Nodle community and builders faster. Ultimately, each Node Operator should be able to have their own Nodle VM stack; thus, both formats could and should be able to co-exist. It will allow builders to use the tools that best fit their needs on a case-by-case basis, but also increase the pace of innovation and experimentation within the Nodle ecosystem. Indeed, new Node Operators could be launched with limited SDK fleets to experiment with new VM implementations or technologies in the future; thus not locking Nodle behind one technical choice.

[10] https://www.antlr.org/
[11] https://webassembly.org/

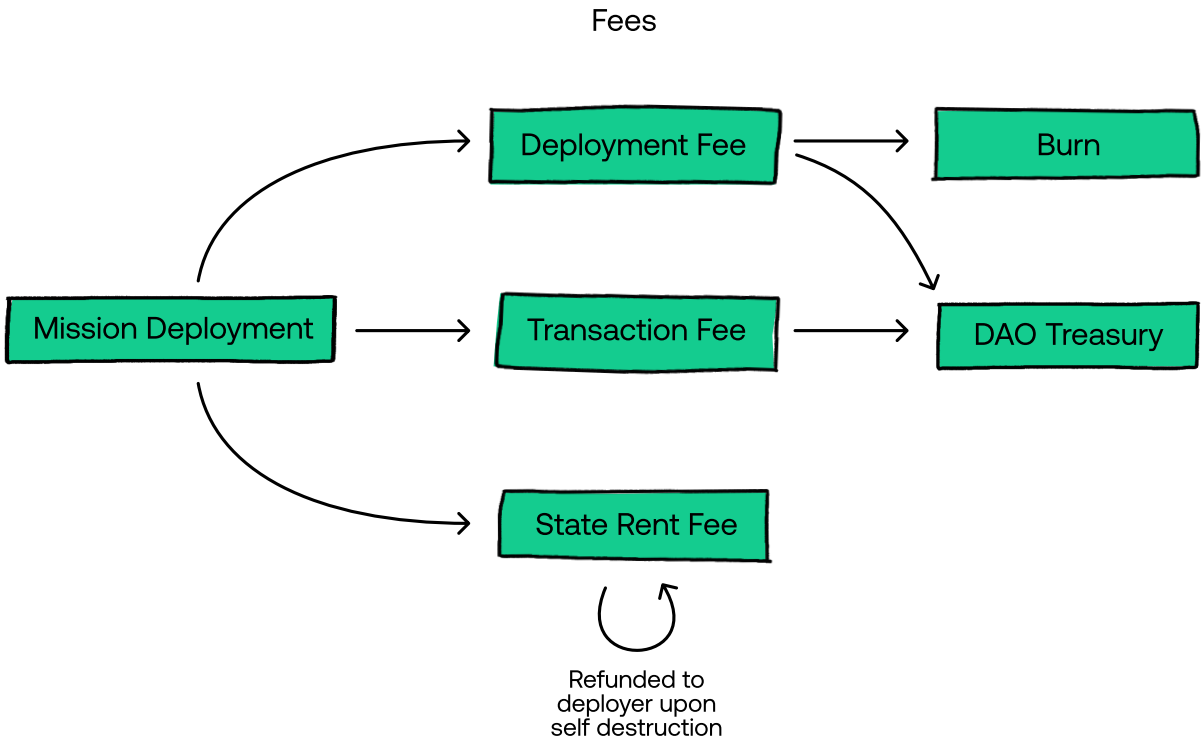
Economics

A Web3 protocol is only as secure as its weakest component. While it is desirable to support the maximum missions possible, this needs to be done in a way that respects the network's capacity. Meaning that the network shouldn't have more missions than it can handle being deployed at any time. Therefore, the following mechanisms must be implemented:

Deployment fee: a dynamic fee system that scales up the cost to deploy a mission depending on the current network usage. A simple solution could be to rely on a constant maximum number of missions and a polynomial function that outputs the fee for the next mission deployment should be sufficient for a first implementation. The NODL used to pay this fee is partially sent to the DAO treasury and partially burned, thus reducing the maximum supply of the network.

State rent: an incentive for builders to "self-destruct" their Smart Missions once they are no longer relevant. This is done by asking them to lock a certain amount of NODL to be refunded to them once the mission's chain code self-destructs. Thus freeing the resources used by the network to support it.

Transaction fees: just like any transaction on the Nodle Parachain, deploying a new Smart Mission or interacting with one will also come at a small cost in the form of NODL tokens which are to be split between the block creator and the DAO treasury.



A Smart Mission must be deployed with its internal incentives to ensure Node Operators and Nodle App users are properly rewarded for their work. If a mission is deployed without internal incentive schemes like tokens or NFT payments, it is likely that it won't be picked up by a Node Operator or Nodle App user.

Examples

The inner workings of the Nodle Missions will be explored through a series of fictional examples which match real-world needs. The beauty of the system lies in its inherent simplicity and flexibility. Indeed, each mission developer can code the guarantees they want for their Smart Missions. For instance, one could decide they are fine with lower trust location proofs, while other builders might ask for multiple proofs or confirmations before unlocking rewards to the mission participants.

The Nodle Network does not provide any strong “proof of X" primitive. Instead, it lets builders define their criteria and set the threshold of what their mission is willing to pay for. Node Operators or Nodle App users can select what they are willing to run on their devices and be compensated appropriately following traditional free market dynamics.

The examples below will focus on highlighting various mechanics of the Nodle Network and its Smart Missions in a simple manner. This means that most production deployments would likely require slightly stronger guarantees or different configurations.

Asset tracking

Asset Tracking Limited (a fictional company for the purpose of this example) wants to find its devices across the city of New York. They are willing to pay 10 NODL for each successful ping from the network. To do so, they deploy a new mission composed of the following components:

- A device code that listens to BLE scans from the Nodle SDK. For each asset found from Asset Tracking Limited, it will send it to the company server and collect a "receipt," which is then submitted to the Nodle Parachain to unlock the appropriate compensation[12].
- A chain code that receives receipts (which are merely signed sequences of bytes) and unlocks the hardcoded compensation of 10 NODL.
- The associated metadata identifies the mission to be part of the asset tracking category and to be restricted to the city of New York.

Asset Tracking Limited will deploy this Smart Mission by deploying the chain code on the Nodle Parachain and paying the appropriate fees described in Economics, and preloading it with enough tokens to support the necessary pings. Right as the chain code gets instantiated, other actors of the Nodle Network perform a series of actions:

01 The Nodle Parachain collators should pin[13] the device code and metadata IPFS documents on their server.

02 The Node Operators listen to the Nodle Parachain and detect that a new mission is instantiated. They fetch the chain code, the device code, and its metadata.

[12] For real-world production use-cases an escrow system might be necessary to lower the trust requirements. For the sake of this paper, the requirements have been simplified to focus on the inner workings of the Missions system itself.
[13] In IPFS terminology, pinning simply means downloading a file to one's local node and making it available to other network participants.

03 Once the Smart Mission is downloaded, the Node Operator will filter it through its proprietary logic to define whether it wants to run it. Typically, it will look for whether they have devices in New York and whether it satisfies its restrictions. Such restrictions might cover which developers are allowed to track which kind of devices.

04 If the Node Operator is willing to support the smart mission, it will look for devices close to or in New York, and load them with the appropriate device code.

Whenever a Nodle SDK detects a device from Asset Tracking Limited as instructed by its Node Operator and the mission's metadata, it will call into the Smart Mission's device code. The device code will perform its normal sequence of operations. In this example, it reports the device identifier, the device's GPS position, and an eventual attestation that it is a genuine Nodle SDK to the company's server. The server then replies with a signed receipt that can then be used to claim the token payment from the chain code[14].

If the mission runs out of tokens to pay out devices, Asset Tracking Limited might reprovision it with additional tokens. The company might also decide to deactivate the mission once they no longer need it or if it runs out of funds. If it so wishes, it would trigger the appropriate transaction which would have the following chain reaction:

01 The chain code would be marked as uninstalled on the Nodle Parachain, thus reducing the state size to be maintained by its collators. If necessary, the chain code might send any funds left back to its deployer, Asset Tracking Limited.

02 The Node Operators listen to the Nodle Parachain and detect that the mission was deactivated. They free the appropriate resources on their side and command their Nodle SDKs to uninstall the mission's device code.

[14] This transaction will most likely go through the Node Operator. The Nodle SDK will notify the Node Operator of the transaction to be sent via an internal API. In the case of the Nodle App, it would send the transaction using the user's wallet, either in the background or after asking the user for its approval.

Geographic airdrops

Modern Coffee LLC (a fictional company for the purpose of this example) wants to create a Smart Mission to attract new customers to their shops around Europe. They do not want to deploy one mission for each shop, instead, they will deploy one global mission that checks the user's positioning and relies on the attestation mechanism described in MobileVM API to claim an NFT voucher which can be redeemed for a free coffee. To prevent users from abusing their marketing program and getting too many free coffees they will impose some restrictions checked by the chain code:

- The user must have been awarded by the Nodle Network 5 NODL, showing that they spent some time running the Nodle App. They want to ensure the 5 NODL were minted as network reward, and thus will not record Nodle tokens sent from another wallet[15].
- The user cannot claim the voucher NFT twice with the same account. To enforce this, the chain code records which account claimed an NFT and prevents double claiming.

[15] To do this, the Nodle Parachain will need to record how many NODL are awarded to each user. This is a feature that is already being planned for a future release of the Nodle Parachain runtime.

Because this is essentially a marketing program, they are satisfied with these guarantees and do not want to burden their customers with higher requirements.

Their Smart Mission deployment is very simple:

01 Their Smart Mission deployment is very simple:

- A simple chain code that receives an attestation that a Nodle SDK instance is genuine. If this attestation is valid, it should double-check how many NODL the caller got awarded previously and whether it already minted a voucher NFT before minting it.

- The device code is a bit more complex: it fetches an attestation that it is running on a genuine device and checks its position. If the device is at a known Modern Coffee LLC location, it sends the attestation to the chain code via a Nodle Parachain transaction.

02 Because the financial upside of a free coffee is rather limited, Node Operators will most likely ignore the mission. However, the mission will be available in the Nodle App where users can choose to install it before or when visiting a Modern Coffee LLC shop. Alternatively, it could be assumed that the mission is downloaded in the background if the application is configured appropriately by its users.

03 Additionally, Modern Coffee LLC locations could display a QR Code with a deep link to auto-install their mission, or embed it in their mobile application if they have one.

Once the marketing program is over, the Smart Mission can be deactivated by Modern Coffee LLC, which returns the state rent they had to deposit when deploying the Smart Mission. Node Operators detect the mission's deactivation and inform the SDKs they manage to uninstall the mission from their VM.

Proof of Participation

Impactful Events Corp (a fictional company for the purpose of this example) is organizing an in-person event in the city of Prague. They want to airdrop a special NFT to each participant. They verify participation at the event through the deployment of custom software on their team's tablets. This software makes the tablets broadcast special Bluetooth beacons, which can be exchanged for an NFT via the mission's chain code.

This example follows a flow similar to the following:

01 Impactful Events Corp deploys a mission composed of the following parts:

- The device code listens to the special beacons from the event, which contain a pre-computed secret from Impactful Events Corp's team. It then sends it to the chain code deployed on the Nodle Parachain.
- The chain code receives these secrets and verifies them. It mints a new NFT as part of the event's collection if they are verified.

02 In this case, most Node Operators will likely ignore the mission because the NFT's financial value is unclear.

03 However, Nodle App users will see the Smart Mission on their application and can choose to install it themselves on their phones if they go to the event.

04 Additionally, Impactful Events Corp can send an email to the participants, or show a QR Code, which contains a deep link to install the mission directly in the Nodle App[16].

05 Alternatively, Impactful Events Corp might own a mobile

[16] In this scenario, it is expected that the Nodle App will ask for the user to confirm the installation of the mission to prevent security and privacy leaks.

application that could embed the Nodle SDK and pre-install their mission.

Once the event is over, Impactful Events Corp will deactivate the mission on the Nodle Parachain, thus freeing some of the NODL they had to lock as part of the state rent defined in Economics, and informing Node Operators to uninstall the mission from the SDKs they manage.

Future tracks

While Smart Missions are a great advancement, they will require additional tooling and integration within the stack to reach their true potential. Indeed, it will be necessary for builders to be able to integrate their frontends with their missions easily. On top of this, the Nodle App will need to become the prime way to interact with any kind of mission and be able to generalize extremely well to support the creativity of mission builders.

Nodle.JS

An easy way to build frontends interacting with the Nodle Network and its Smart Missions will be required to enable Smart Mission builders to create their frontends. Custom frontends are necessary to support builders in creating unforeseen use cases.

Similarly to Web3.JS, Ethers, or Polkadot JS, a client library tentatively named Nodle JS will need to be developed. It will be a small library for front-end developers to easily interact with the network via their web or CLI applications. Additionally, Nodle JS will need to integrate with the user's wallet extensions such as Talisman or Polkadot JS to be able to send transactions to the Nodle Parachain.

Mission browser in Nodle App

The Nodle App will need to support an in-app browser so users can also interact with these more advanced Smart Missions through their preferred mobile Nodle wallet. It will be required for the app to inject a Polkadot JS-compatible object to allow the front end to ask for transactions to be sent to the Nodle Parachain.

This is similar to Web3 JS integrations featured by Metamask Mobile, Ledger Live, and many other web3 applications.

Mission libraries

As more builders join the ecosystems and create Smart Missions, common needs and implementations will become easier to identify. Once this is possible, some teams could develop a common library of mission building blocks that make it even easier to write a mission with various libraries vetted by the community to generalize well, be optimized, and free of bugs.

This is similar to OpenZeppelin on Ethereum, or Substrate FRAME for Polkadot.

About Nodle
Nodle is a programmable network powered by smartphones, connecting the physical world to the internet and enabling unique geolocation applications.
Visit us at nodle.com.

nodle