Ocelloids Network Specification

The Zone Council, SO/DA

February 15, 2024

This document is a **work in progress**.

Abstract

This document outlines the functional specifications of the Ocelloids network, a system for secure and transparent leasing of software agents that observe, correlate, and respond to blockchain activity. The network comprises provider nodes hosting agents under negotiated leases with consumers. A continuous service attestation process ensures reliable agent operation, while the network registry controls the admission, suspension and termination processes for service providers and auditors within a trust zone. The document details network roles, leasing processes, and attestation methods, providing a comprehensive understanding of the Ocelloids network's functionalities.

1 Introduction

The Ocelloids network is as a real-time reactive layer comprising software agents that observe, correlate, and respond to blockchain activity. These agents are available in a marketplace, where consumers can host them on provider nodes through a leasing model. The agent's positioning is ideal for connecting offchain data with onchain activity. Providers undergo an admission process within a trust zone and operate within defined geographical boundaries, subject to applicable laws. The Service Attestation process ensures the reliable functioning of the agents, enabling providers to earn leasing fees for hosting them.

2 Network

An Ocelloids network is a collection of nodes (section 3) under a common trust zone with vetted access for participation.

The network is governed through an onchain registry in the main chain, known as the Network Registry, containing participant information and their Network Roles.

Network participants have well-known identities tied to specific organizations, geographic locations, blockchain accounts[1], and publicly accessible endpoints. All actions carry a valid digital signature from the originating participant's key pair associated with a registered account, enabling verification.

2.1 Network Registry

The network registry exists in the main chain and delineates a trust zone characterized by designated sovereign and administrative accounts. These accounts are endowed with privileges related to the admission, suspension and termination of service providers and auditors in the network. These procedures play a critical role as they establish connections with entities subject to liabilities and potentially involve the establishment of legally binding contractual agreements.

Each participant of the network has an associated Party Record, which includes essential information for locating service providers and verifying the authenticity of digital signatures.

[1] A blockchain account address is typically derived from the public key of a cryptographic key pair. Usually, this public key is recoverable from a digital signature, given the signed content.

Table 1: Party Record

Property	Description
Subject	Distinguished name[1].
Accounts	Operator[2] and vault account[3] addresses.
Endpoints	URLs[2] of the public endpoints.
Location	Geographic point location[3].
Role	The node role (section 2.2).

2.2 Network Roles

The network encompasses two node roles.

Provider Node
: Negotiates service agreements, hosts agents in accordance with active leases and receives leasing fees.

Auditor Node
: Verifies the execution of the agents involved in active leases and provides attestations to authorize the payment of leasing fees.

3 Nodes

An Ocelloids node is a logical entity that could be deployed on multiple physical systems, e.g. horizontally scalable clusters. Essentially, a logical network node provides:

1. Ingress services for sourcing onchain data.
2. Runtimes for program execution.
3. Remote endpoints to serve client connections.
4. Egress services for data emission and notification.

Reference Ocelloids Node / Runtime spec. when written.

[2]The operator account actively interacts with the blockchain, anchoring hashes, providing attestations, etc.

[3]The vault account accrues fees and its private key could be kept in cold storage.

4 Programs

4.1 Program Catalog

The Ocelloids network maintains a curated catalog on the main chain of programs eligible for hosting. These programs are organized into packages identified by content-addressable identifiers and stored in a content-addressable storage network. Each package contains source code and/or binary files, along with a manifest describing the program and its required hosting capabilities. An instance of a program is referred to as an agent and possesses a unique agent identifier.

4.2 Program Execution

When a program instance—or agent—is hosted by an Ocelloids node, the node's runtime fulfills its execution according to the program's manifest. The runtime provides necessary resources and sandboxing mechanisms to ensure isolation and security. Programs interact with the node's ingress and egress services to access onchain data and emit data streams.

5 Service Leasing

The service leasing process involves consumers requesting service offers, depositing funds to pay for agent hosting, undergoing continuous service attestations, managing periodic payment claims, and facilitating automatic lease renewals.

5.1 Service Agreement

The service agreement process mandates that the consumer deposit funds to cover at least one period before the provider provisions the agent. The consumer queries the Programs to determine the identifier of the program it wishes to execute. The providers are discovered through the Network Registry. The high-level steps are as follows.

1. *Request Quote*. The consumer initiates the leasing process by submitting a quote request to the provider, specifying the desired program identifier for execution.

2. *Service Offer.* The provider responds with a service offer, providing details such as the program identifier to be executed, the leasing period duration in number of blocks, leasing fee, and minimum deposit required.

3. *Place Deposit.* The consumer submits a deposit to the main chain, specifying the offer and the transfer amount. The funds for one leasing period are locked, with any remaining funds available for withdrawal by the consumer at any time.

4. *Confirm Deposit.* The main chain issues a deposit receipt for the offer to the consumer, confirming the deposit. The consumer then sends this receipt to the provider.

5. *Provision Agent.* The provider verifies the deposit and provisions the agent based on the accepted offer.

6. *Confirm Lease.* The provider submits the deposit receipt to the main chain to formalize the lease, receiving a lease receipt in response.

7. *Activate Lease.* The provider acknowledges the lease activation to the consumer upon receiving the lease receipt.

5.2 Service Attestation

The service attestation process[4] involves auditors continuously verifying the accurate operation and fulfillment of the agents hosted by a providers under the leasing duration. The attestation process operates within the timeframe of a leasing period. The hosted agent observes the activity of monitored chain/s, processing each block, which constitutes the primary input of on-chain data. The provider commits a verifiable proof of the processing of each block to a locally maintained verifiable map. Since blocks could be processed out of order, the provider maintains a local verifiable key-value map independent of the insertion order, such as a sparse Merkle tree[4]. The commitment to the map adds a pair (k, v), where $k = BlockHash$ and $v = \text{digest}(ProgramOutput)$. The provider must anchor the top hash of the verifiable map to the main chain at the end of the period. The attestation process for each period works as follows.

[4]For non-deterministic sources affecting the program output, a snapshot mechanism must be provided for reproducibility.

1. *Request Service Proofs.* The auditor requests service proofs for a random sample[5] of block hashes R from monitored chain within the most recent anchored period.

2. *Present Service Proofs.* The provider presents the requested inclusion proofs for the given monitored chain block hashes. $P_R = \{Proof(b_H) : b_H \in R\}$, where $Proof$ produces an inclusion proof for the commited value v on the key b_H.

3. *Verify Service Proofs.* The auditor verifies the inclusion proofs P_R.

   (a) Confirms the inclusion proof using the anchored top hash for the period retrieved from main chain.

   (b) Independently processes the selected blocks, retrieving them from monitored chain[6], to verify that the digest of the resulting program output matches the value v of the requested block in the proof.

4. (i) **On successful verification**
   Record Attestation. The auditor submits a signed attestation of the verified period. The attestation authorizes[7] the payment of leasing fees[8] by the provider.

   (ii) **Otherwise**
   Record Dispute. The auditor submits a signed dispute with details for further resolution[9].

This continuous attestation process ensures the maintenance of verified operational records, serving as a prerequisite for claiming leasing fees.

5.3 Payment & Lease Renewal

At the end of each leasing period, the provider must claim the leasing fees and an automatic renewal process is initiated.

[5] A simple approach would be to use Yamane's method ($n = \frac{N}{1+Ne^2}$) for N blocks in the period, where $n \approx 400$ for a 1-month period with 95% confidence.

[6] We assume that the historical blocks of monitored chain/s for a leasing period are retrievable at the end of the period.

[7] Variations could require signatures from multiple auditors.

[8] The authorized payment should be captured by the provider and could entail the deducation of a managment fee accrued to the auditor/s.

[9] While operating in a vetted governance model (i.e., without funds at stake), the resolution falls under the discretion of the trust zone sovereign entities, who are responsible for suspending or terminating the misbehaving party and transferring funds to the rightful party.

1. *Claim Fees.* Provider claims the payment of the fees for the leasing period by submitting a transaction to the main chain; e.g., *ClaimPaymentForPeriod*.

2. *Verify & Transfer.* Main chain verifies the latest agent operational attestation and transfers the funds to the provider.

3. *Renewal Check.* Provider checks renewal conditions.

   (a) The lease is active.

   (b) Sufficient funds for next period in the consumer account.

4. (i) **On successful check**
   Renew Lease. Main chain initiates the renewal process, locking funds for the next period.

   (ii) **Otherwise**
   Cancel Lease. Main chain cancels the lease, and the provider takes appropriate actions to decommission the agent.

This process ensures a smooth transition between leasing periods, with automatic renewals and the flexibility to cancel if necessary. Consumers are responsible for maintaining sufficient deposit funds to cover renewals.

Glossary

agent
: A program instance running on a provider node. 2–5, 7, 8

agent identifier
: A unique identifier for a program instance running in a specific host. 4

auditor
: Node responsible for verifying the execution of agents involved in active leases and providing attestations to authorize the payment of leasing fees. 3, 5, 6

consumer
: Party that leases the hosting of an agent on a provider within the network. 2, 4, 5, 7

main chain
: Blockchain that provides the governance, discovery, leasing, anchoring and settlement functions for an Ocelloids network. 2, 4–7

monitored chain
: Blockchain that serves as onchain data source for an agent. 5, 6

program identifier
: A unique content-bound identifier pointing to a program package stored in a content-addressable storage network. 4, 5

provider
: Node responsible for negotiating service agreements, hosting agents according to active leases, and receiving leasing fees. 2–8

References

[1] The directory: Selected attribute types. X.520, October 2019. URL https://handle.itu.int/11.1002/1000/14037.

[2] Tim Berners-Lee, Roy T. Fielding, and Larry M Masinter. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986, January 2005. URL https://www.rfc-editor.org/info/rfc3986.

[3] Standard representation of geographic point location by coordinates. ISO 6709:2022, September 2022. URL https://www.iso.org/standard/75147.html.

[4] Rasmus Dahlberg, Tobias Pulls, and Roel Peeters. Efficient sparse merkle trees: Caching strategies and secure (non-)membership proofs. Cryptology ePrint Archive, Paper 2016/683, 2016. URL https://eprint.iacr.org/2016/683. https://eprint.iacr.org/2016/683.