ALLWIT

A KEY, NOT AN IDENTITY

EVERY MODEL. NOBODY WATCHING.

Allwit: Private, Wallet-Native Access to Frontier and Open AI, with a Hub to Run Open Models and a Studio to Build Agents

OCTOBER 28, 2026 V1.5 ALLWIT.AI

ABSTRACT

Allwit is a platform for using, sharing and building with AI models without handing over an identity. It joins three surfaces that today exist as separate products: a gateway to frontier and open models across text, image, video and speech; a *Hub* where open-weight models, adapters, datasets and agent templates are published and, unlike a plain repository, can be run in place; and an *Agent Studio* for building agents from any model in the catalogue without code. All three rest on one wallet identity, one balance of WIT credits and one router. Users sign in with an Ethereum wallet rather than an email address. WIT credits are granted every day to wallets that stake $ALL, an ERC-20 token on Ethereum, as a function of the $ALL staked and the $ALL price, or are bought with USDC, USDT or $ALL; either way they are spent as blind-signed tokens that cannot be linked back to the wallet that obtained them. From launch, half of all platform fees are used to buy $ALL back from the market every day, and half of every $ALL paid for credits is burned at the end of each quarter. Requests travel through an independent oblivious relay to a gateway that keeps no prompt logs, so that no single party, Allwit included, observes both who asked and what was asked. Each model carries one of two privacy labels: *Private*, for models served on hardware Allwit controls, where no outside party sees the prompt; and *Blind*, for third-party providers, which see the prompt but not the person. An agent inherits the weakest label among the models and tools it uses. The platform ships in four phases of rising assurance: Gateway (frontier and open models behind wallet sign-in, credits and blind routing), Hub and Agent Studio, Open Network (independent relays, a private compute tier and an attested no-log gateway), and Native (Allwit's own router and models, and agents that hold wallets). This draft specifies the architecture, privacy model, $ALL tokenomics, the WIT credit mechanism, the $ALL buyback and burn, and the phase plan. The financial model is deferred to a companion paper; where this draft depends on it, the dependency is stated and the value is left open.

INDEX

CONTENTS

1	Introduction
2	Background and related work
3	System overview
4	Identity, $ALL and WIT credits
5	Blind routing and privacy tiers
6	The model gateway
7	Allwit Hub
8	Agent Studio
9	Phase 1: Allwit Gateway
10	Phase 2: Hub and Agent Studio
11	Phase 3: Open Network
12	Phase 4: Native
13	Responsible use without identities
14	Trust progression and security summary
15	Market design and positioning
16	Governance
17	Limitations and open problems
18	Conclusion
A	Parameter reference
B	Anatomy of a sealed request
C	Open questions and future work
	References

§ 01

INTRODUCTION

Generative AI has fragmented across many strong providers and a fast-growing open-weight community. No single model is best at every task, and the most useful applications increasingly combine several models into agents. A serious user therefore holds accounts, keys and subscriptions with several providers, each of which binds a name, an email address and a payment card to a complete history of prompts. That history can be stored, analysed, breached or compelled. Aggregators such as OpenRouter [1] have removed the first cost, the fragmentation of keys and bills, but not the second: the aggregator itself now observes both who the user is and what the user asks, and a card is still required to pay.

Two adjacent gaps compound the problem. Open-weight models are published in the thousands on hubs such as Hugging Face [2], but most people can browse them and cannot run them: using one means downloading gigabytes of weights, renting accelerators and operating a server. And agent builders are typically bound to one vendor's models, tools and billing, so an agent cannot use the best model for each step or move when a better one appears.

Allwit addresses these gaps in one platform built on a single premise: *access to AI should require a key, not an identity.* The key is a wallet. The wallet stakes $ALL and receives WIT credits; the credits are converted into unlinkable tokens; the tokens pay for requests that reach models through a path on which no party sees both ends. The same wallet, balance and path then extend to running community models and to building agents.

1.1 CONTRIBUTIONS

- **A unified surface.** One OpenAI-compatible API [3] and one chat application over frontier and open models in four modalities, with routing, automatic fallback and a single meter (§6).
- **Identity separated from content.** Wallet sign-in [9], blind-signed credit tokens [4], [5] and an oblivious relay [8] compose so that the credit issuer, the relay, the gateway and the model provider each hold at most one of the two facts that matter: who asked, and what was asked (§5).
- **Access priced by stake, not by identity.** A wallet's daily WIT credits are a function of the $ALL it has staked and the $ALL price; the stake is never consumed, and unused staking credits expire each day, so the platform never carries a staking balance it owes. Credits can also be bought outright, with stablecoins or with $ALL (§4).
- **Honest privacy labels.** Every model, Hub listing and agent carries a *Private* or *Blind* label stating which outside parties can read the prompt; labels compose by taking the weakest (§5.4, §8.4).
- **A runnable hub.** A catalogue of open models, adapters, datasets and agent templates in which publishing and serving are separate steps, so the catalogue can be broad while the served set stays fast; creators are paid from usage (§7).
- **Vendor-neutral agents.** A no-code studio whose agents may mix models per step, keep their state encrypted to the owner's wallet and run under explicit tool permissions and spend caps (§8).

- **A phased path to verifiable claims.** A four-phase plan in which each phase is a complete product and each later phase replaces an operational promise with a mechanism: independent relays, a private compute tier and a gateway whose no-log property is attested (§9–§12, §14).

1.2 DESIGN PRINCIPLES

1. **Separate who from what.** No component that learns a user's identity, network address or wallet should also learn the content of that user's requests, and the converse.
2. **Label rather than overclaim.** Where a third party must see the prompt, the platform says so per model, instead of describing the whole system as private.
3. **One foundation, three surfaces.** Models, Hub and Agent Studio share identity, credits, routing and labels. A feature that needs its own account or balance is out of scope.
4. **Each phase ships alone.** Phase 1 is a complete product. Later phases add to the same wallet, balance and API without changing how users sign in or pay.
5. **Promises become mechanisms over time.** Early phases rely on stated operational commitments. The plan names each one and the later mechanism intended to replace it.

1.3 WHY NOT EXTEND AN EXISTING AGGREGATOR

An aggregator that adds a privacy setting retains the property that defeats it: a single operator that both bills an identified customer and relays that customer's prompts. Blind routing is a property of the whole request path, including how payment is represented, and cannot be retrofitted as an option. The same holds for the Hub and the Studio: a hub that cannot run what it lists, and an agent builder bound to one vendor, are each limited by the account and billing model underneath them. Allwit therefore begins from the payment and identity layer and builds the surfaces on top.

1.4 SCOPE OF THIS DRAFT

This paper specifies architecture, privacy model, surfaces, phasing, the $ALL token and its distribution, and the mechanism by which staked $ALL becomes WIT credits. It does not specify the financial model: the thresholds of each staking tier, the sizes of credit packages, creator reward shares, relay compensation, vesting schedules, or the sizing of the Allwit Compute Treasury against usage. Those are the subject of a companion paper. Where a mechanism here depends on one of those values, the value is marked `TBD` and collected in Appendix A.

---

§ 02

BACKGROUND AND RELATED WORK

Model aggregators. Aggregators expose many providers behind one API, typically OpenAI-compatible [3], and add routing and fallback [1]. Learned routing between strong and weak models can reduce cost at small quality loss [24], [25]. Aggregators today are account-based and card-funded, and see both customer

identity and prompt content.

Model hubs. Public hubs [2] host weights, model cards and datasets, and have standardised safe serialisation formats such as safetensors [16]. Hosted inference on a hub is generally limited to a curated subset, and creators are rarely paid from downstream use.

Efficient multi-model serving. Paged attention [15] and batched multi-adapter serving [14] make it economical to serve many low-rank adapters [13] over a shared base model. Allwit relies on this to offer community fine-tunes at marginal cost (§7.3).

Anonymous credentials and oblivious transport. Blind signatures [4] let an issuer sign a token without seeing it, so that later redemption cannot be linked to issuance. Privacy Pass standardises issuance and redemption of such tokens [5], [6], including RSA blind signatures [7]. Oblivious HTTP [8] splits a request between a relay that sees the client's address and a gateway that sees the content. Allwit composes these primitives rather than introducing new ones.

Confidential computing. CPU trusted execution environments [21], [22] and GPU confidential computing [20] allow a remote party to verify, through attestation, which code is running. Apple's Private Cloud Compute [26] demonstrated a production design in which a no-retention claim about AI inference is made checkable by publishing attested software images. Allwit's Phase 3 gateway follows the same approach (§11.3).

Wallet identity. Sign-In with Ethereum [9] defines a signed message that proves control of an address for a session, and WalletConnect [12] connects mobile and desktop wallets to applications. Account abstraction [11] enables programmable wallets with spending policies, which Allwit uses for agent wallets (§8.7).

Stake-for-access designs. Several networks grant usage rights to holders who lock a token rather than spend it, so that access is a deposit and the token's utility rises with demand for the service. Allwit follows that pattern for $ALL, with one addition: the daily grant is denominated in the stake's dollar value, so that what a staker receives in compute tracks the market value of what they have locked (§4.4).

Decentralised inference networks. Networks such as Bittensor [27] incentivise open participation in model serving. Their focus is the supply side and its incentive design; Allwit's focus is the demand side, the user's identity and privacy, and it can treat such networks as one possible source of compute for the Private tier, subject to the trust model in §14.

---

§ 03

SYSTEM OVERVIEW

3.1 THREE SURFACES

Models is the gateway: frontier and open models across text, image, video and speech, behind one chat application and one API, with Allwit's native models later (§6). **Hub** is the catalogue of community models, adapters, datasets and agent templates; any served listing appears in the model picker and the API (§7). **Agent Studio** builds agents from catalogue models, tools, knowledge, memory and triggers, and can publish them to the Hub as templates (§8).

3.2 THE SHARED FOUNDATION

Four components are common to all three surfaces. *Wallet identity* signs in, stakes, publishes and owns agents (§4.1). *One credit balance* of WIT credits pays for every model call, Hub run and agent step (§4.2). *Blind routing* separates the requester from the request (§5). *One router* receives every request, whether typed in chat, sent through the API or issued by an agent, selects a model, falls back on failure and meters usage (§6). Figure 1 shows the arrangement.

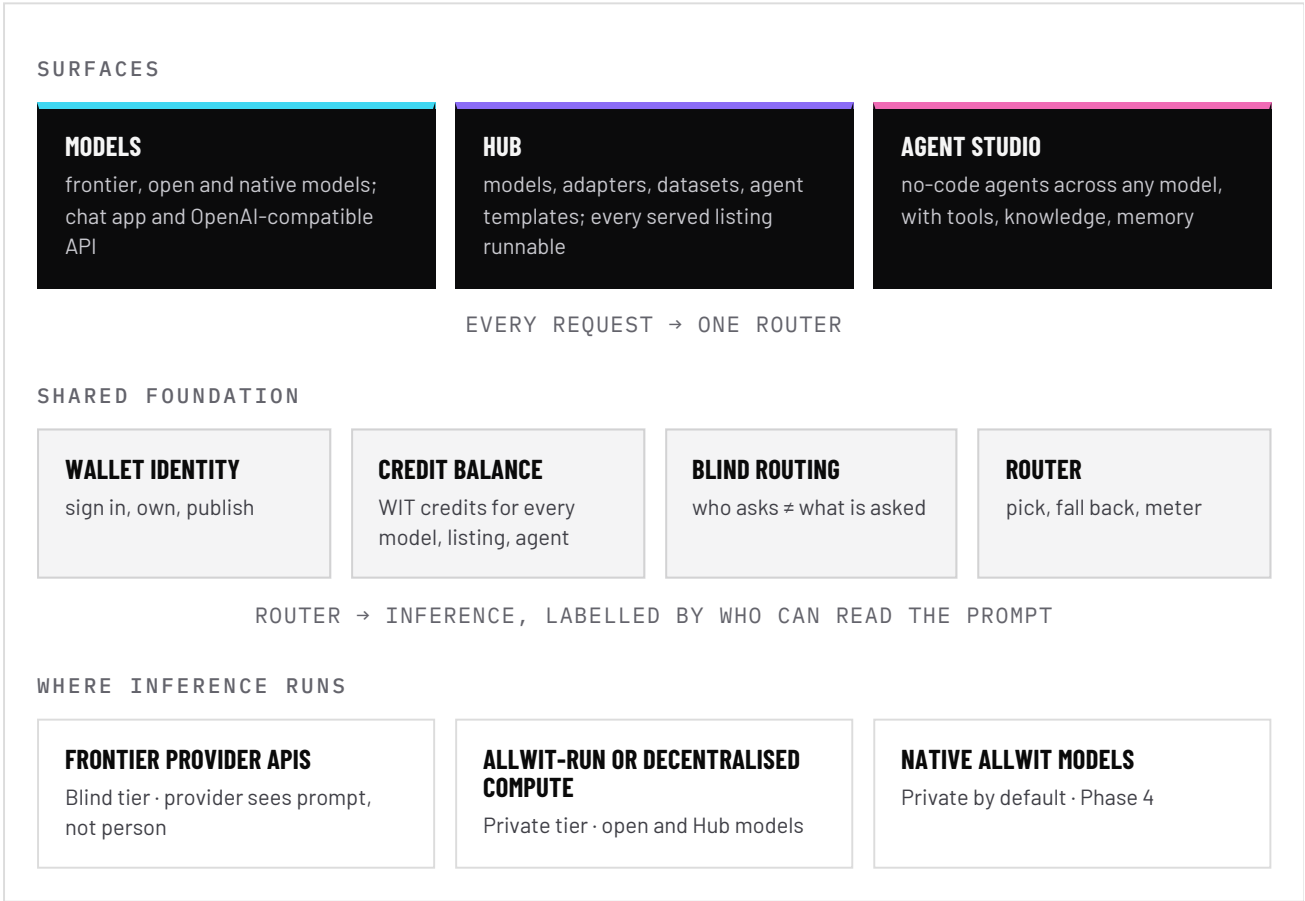

FIGURE 1: System overview. Three surfaces share one identity, balance, routing path and router. Inference runs either at a third-party provider (Blind tier) or on hardware Allwit controls (Private tier).

3.3 PARTICIPANTS

- **Users** hold a wallet, obtain WIT credits and send requests through the app or API.

- **Builders** create agents in the Studio; **publishers** list models, adapters, datasets and templates on the Hub. Both are identified only by wallet.
- **The credit issuer** reads each wallet's stake, the day's $ALL price and any credit purchases, computes the wallet's WIT credits and blind-signs credit tokens for them (§4.7).
- **The Allwit Compute Treasury** holds a stablecoin reserve and the protocol's own $ALL allocation, and receives half of every credit purchase made in $ALL; it pays model providers for the compute that WIT credits buy and funds sponsored packages (§4.3, §4.6).
- **The relay** forwards encrypted requests and sees client network addresses but not content. In Phase 1 it is operated by an independent party; in Phase 3 by several, including $ALL stakers (§11.1).
- **The gateway** decrypts requests, redeems tokens, applies screening, routes to a model and meters usage. It never learns a wallet or network address.
- **Model providers** serve Blind-tier models under commercial agreements with Allwit; **compute operators** serve Private-tier models.

3.4 THE PHASE LADDER

Table 1 summarises what each phase adds and which trust it removes. Phases are cumulative. Dates are not yet set.

PHASE	ADDS	PRIVACY MECHANISM	RESIDUAL TRUST
1 · Gateway	Frontier and open models, four modalities, wallet sign-in, WIT credits	Blind tokens; single independent relay; no-log gateway by policy	Gateway operator's no-log commitment; relay and gateway not colluding
2 · Hub + Agent Studio	Runnable Hub listings, no-code agents, creator rewards	Owner-encrypted agent state; label composition	As Phase 1; publishers' licence declarations
3 · Open Network	Independent and staker-run relays, private compute tier, attested gateway	Multiple relay operators; TEE-attested no-log gateway	TEE vendor and attestation chain; Blind-tier providers' own policies
4 · Native	Native router and models, agents with wallets	Native models served Private by default	As Phase 3; agent wallet policy contracts

TABLE 1: Phase ladder. Each row lists the residual trust a user still extends at that phase; §14 restates these assumptions in full.

§ 04

IDENTITY, $ALL AND WIT CREDITS

4.1 WALLET IDENTITY

A user's identity on Allwit is an Ethereum address. The client connects a standard wallet, by browser extension or through WalletConnect [12], and signs a Sign-In with Ethereum message [9] that binds the address to a short-lived session with a nonce, domain and expiry. No name, email, phone number or card is collected. The same address stakes $ALL, buys credits, publishes to the Hub and owns agents, so credits, listings and agents follow the key. There is no password database, and account recovery is the wallet's own.

The session proves the right to obtain credits. It is deliberately *not* attached to inference requests: the wallet is used at issuance and never at redemption (§4.7).

4.2 TWO UNITS: $ALL AND THE WIT CREDIT

Allwit separates two units with different jobs.

- **$ALL** is an ERC-20 token [10] on Ethereum mainnet with a fixed supply of 1,000,000,000. Its platform role is access: staked $ALL entitles a wallet to a daily grant of WIT credits for as long as it remains staked. Staked $ALL is not consumed by use, and unstaking returns the full amount after a cooldown. $ALL can also be spent on WIT credits outright, converted at its current price (§4.5).
- **The WIT credit** is the platform's unit of account for usage. Every model, Hub deployment and agent step has a price specified in WIT credits, set from its underlying cost, so a short text completion costs little and a video clip costs much more. WIT credits live in the Allwit platform ledger, are not transferable between wallets, and are kept to two decimal places. Credits are sold at a retail price of $0.01 each: 100 WIT credits for 1 USDC or 1 USDT.

Keeping the units apart separates the two things that move: a credit's price for a given model tracks that model's cost, and is adjusted as provider charges change; the number of credits a wallet receives each day is a function of its stake and the $ALL price (§4.4). A user therefore reads prices in one stable unit and sees their allowance rise and fall with the token they hold.

4.3 $ALL: SUPPLY, DISTRIBUTION AND THE COMPUTE TREASURY

$ALL is issued once, with a fixed supply of one billion, and distributed as in Table 2. Forty percent of the supply, 400,000,000 $ALL, is sold to the public through a 90-day auction before the token generation event (TGE), open to anyone, so the price is set by demand rather than by private rounds. The auction offers a daily allocation of about 4.44M $ALL (400M ÷ 90) under five rules:

- **Floor price.** Day 1 opens at a floor price of $0.05 per $ALL. Each day has a single price, at which every token sold that day is sold; that price is the day's clearing price.
- **Daily price step.** If 50% or more of the tokens on offer on a day are sold, the next day's price is 1% higher. If less than 50% are sold, the next day's price stays the same.

- **Carry-over.** Tokens unsold at the end of a day are added to the next day's allocation and offered at the next day's price: 1% higher after a price step, the same otherwise.
- **Close.** Tokens still unsold after day 90 are either burned or transferred to the Allwit Compute Treasury. Which of the two applies is not yet decided (Appendix A).
- **DEX pool.** The auction's final clearing price sets the opening price of the Uniswap pool created with the protocol-owned liquidity allocation.

Because the price rises by at most 1% a day, the day-90 price is at most about $0.121. If every day's allocation sells out, the auction raises about $32.2M at an average of about $0.080 per $ALL. Selling every token at the floor would raise $20M, and the raise cannot exceed about $48.5M, every token at the day-90 maximum. On the full supply of one billion, a final clearing price between $0.05 and $0.121 implies a fully diluted valuation between $50M and $121M. Half of the raise is placed in the Compute Treasury and half funds development and operations.

ALLOCATION	% OF SUPPLY	$ALL	PURPOSE
Public auction (90 days, pre-TGE)	40%	400,000,000	Sold from a $0.05 floor in daily allocations of about 4.44M; buyers stake it for WIT credits; proceeds fund the Compute Treasury and development
Community rewards (incl. genesis airdrop)	28%	280,000,000	Genesis airdrop and ongoing rewards for the Allwit community
Core contributors	13%	130,000,000	Team, subject to vesting
Allwit Compute Treasury	10%	100,000,000	Staked by the protocol to fund sponsored free and basic packages
Foundation	6%	60,000,000	Long-term development of the protocol
Protocol-owned DEX liquidity	3%	30,000,000	Paired with ETH or USDC in a Uniswap pool opened at the final auction clearing price
Total	100%	1,000,000,000	

TABLE 2: Distribution of the $ALL supply. Vesting schedules for contributors, the foundation and community rewards are set in the financial model (Appendix A).

Model providers bill Allwit in fiat, so the compute behind WIT credits is paid for off-chain by the **Allwit Compute Treasury**. It has two parts:

- **Stablecoin reserve.** Half of the public auction raise is held in stablecoins and set aside to pay providers for the compute that WIT credits buy. Credit purchases paid in USDC or USDT are also applied to provider, infrastructure and development costs (§4.6).

- **$ALL allocation.** Ten percent of the supply is allocated to the treasury and staked by the protocol under the same rules as any wallet. The WIT credits it receives fund the sponsored free and basic packages through which new users try Allwit before they hold any $ALL. The treasury stakes its allocation in stages, starting with about a quarter in year 1. Half of every credit purchase paid in $ALL is added to the treasury and used to pay providers, infrastructure and development (§4.6).

Because staking credits expire daily and are recomputed from the live price (§4.4), the treasury never carries a staking balance owed to users: it funds compute as it is used, not credits that might be used later. Purchased credits are paid for in advance.

4.4 THE DAILY WIT GRANT

At every daily reset the issuer grants each staking wallet a fresh balance of WIT credits. The grant depends on two numbers only: the total $ALL the wallet has staked, and the $ALL price that day.

WIT CREDITS PER DAY = F ($ALL STAKED , $ALL PRICE)

The grant is proportional to the stake: a wallet that stakes twice as much receives twice the credits. And it rises and falls with the price: because AI is bought in dollars, the grant is defined on the dollar value of the stake first and converted into credits second, so a staker holds a claim on a share of the platform's compute sized by what their stake is currently worth. A rising $ALL price gives every staker more compute; a falling one gives less; the token's market value and its utility move together. Four properties follow.

- **Granted, not minted.** Nothing is borrowed and nothing is owed. Credits are an entitlement that the stake earns each day, recorded in the platform ledger; unstake, and the grants stop.
- **Dynamic.** The grant is recomputed every day. If $ALL doubles, tomorrow's credits double; if it halves, they halve. The price used is a time-weighted average from an on-chain oracle rather than the spot price, so a single day's swing does not move allowances (Appendix A).
- **Daily.** Unused staking credits expire at the reset. Allowances cannot be accumulated and spent in a burst, and Allwit never holds a staking liability to its users. Purchased credits carry over (§4.5).
- **One price list.** A request costs the same number of credits whichever way the credits were obtained: by staking, by purchase, or through a sponsored package.

4.5 OBTAINING CREDITS

Staking. A wallet locks $ALL in the Allwit staking contract on Ethereum and receives its WIT grant every 24 hours while the stake remains locked. Unused staking credits expire at the next reset. Unstaking returns the full amount after a cooldown. Staking packages, from an entry-level tier to a team tier, make allowances legible; their thresholds are set in the financial model and marked `TBD` here.

Buying credits. Users who prefer not to stake, or who need more on a given day, buy WIT credits at the retail price of $0.01 each, as a top-up or a package. Paid in USDC or USDT, 1 unit buys 100 credits. Paid in $ALL, the amount is converted at the oracle price at the time of purchase, so 1 $ALL buys 100 credits when $ALL is at $1 and 10 credits when it is at $0.10. Purchased credits carry over from day to day. Package sizes are TBD. How each payment is used is set out in §4.6.

Sponsored packages. Free and basic packages, funded by the Compute Treasury's own stake (§4.3), let a wallet try the platform before it holds any $ALL.

Credits from every source are drawn into the same balance and are indistinguishable once converted into tokens. Credits that expire soonest are spent first, so a day's staking allowance is used before purchased credits.

4.6 BUYBACK AND BURN

From launch, half of all platform fees are used to buy $ALL back on the secondary market every day. Half of every WIT credit purchase paid in $ALL is burned at the end of each quarter; the other half goes to the Compute Treasury. Purchases paid in USDC or USDT pay model providers, infrastructure and development. Table 3 sets out each flow.

SOURCE	USE	EFFECT ON $ALL
Platform fees	50% buys back $ALL on the secondary market every day, from launch	Daily market purchase of $ALL
$ALL	50% goes to the Compute Treasury, which pays model providers, infrastructure and development	The other 50% is burned at the end of each quarter
USDC or USDT	Pays model providers, infrastructure and development	No buyback or burn

TABLE 3: Buyback and burn by source. Credits from staking and sponsored packages are outside these flows.

A purchase of 1,000 $ALL at a price of $0.10 buys 10,000 WIT credits: 500 $ALL goes to the Compute Treasury, and 500 $ALL is set aside and burned at the end of the quarter. A purchase of 100 USDC also buys 10,000 credits and is applied to providers, infrastructure and development.

The two mechanisms tie demand for $ALL to use of the platform. Platform fees fund a market purchase of $ALL every day, and each credit purchase paid in $ALL removes $ALL from circulation. Because the supply of one billion is fixed and never increased, every burn reduces it permanently. Buybacks and burns are Ethereum transactions, so both are publicly recorded.

4.7 BLIND CREDIT TOKENS

Credits are spent as anonymous tokens following the Privacy Pass architecture [5]. Once per allowance period the client authenticates as its wallet to the credit issuer, generates a batch of random token values, blinds them and submits them for signing. The issuer checks the wallet's entitlement for the day, signs the

blinded values with a key for the relevant denomination [6], [7], and records only that the entitlement was used. The client unblinds the signatures and holds valid tokens the issuer has never seen.

At request time the client attaches one or more tokens to the sealed request (Appendix B). The gateway verifies each signature and checks a spent-token set to prevent double redemption. Because the issuer signed blinded values, neither it nor the gateway can link a redeemed token to the issuance session, and therefore to a wallet. Unlinkability holds across the population of wallets issued under the same key in the same period, so the design requires few issuance keys per period and batched, regular issuance (§17).

Denominations and change are handled by issuing tokens in a small set of fixed values and returning a signed refund token when a request costs less than the tokens presented. Refund tokens are issued blind by the gateway under the same construction. Because a day's grant varies with the $ALL price, the issuer rounds each grant to the available denominations and carries the remainder within the day's ledger balance, never across the reset.

4.8 WHAT TOUCHES ETHEREUM

Only coarse, infrequent, public actions are on chain: staking and unstaking $ALL, credit purchases settled in USDC, USDT or $ALL, the daily $ALL buybacks from platform fees and the quarterly $ALL burns. Issuance, redemption and metering are off chain, as is the daily grant itself, which the issuer computes from the on-chain stake and the oracle price. This is a privacy choice as well as a cost one. A stake is public by nature; inference is not, and recording it on chain would publish exactly the link the rest of the design removes. The cost of Ethereum mainnet transactions for small stakers is acknowledged as an open problem (§17).

4.9 DEFERRED: THE FINANCIAL MODEL

> COMPANION PAPER
>
> The following are fixed in this draft only as mechanisms. Their values will be specified in the financial model, and this section will be revised to reference it.

- **Tier thresholds.** The $ALL stake that defines each package tier, and the unstaking cooldown.
- **Credit packages.** Package sizes for purchased WIT credits. The retail price, $0.01 per credit, and carry-over of purchased credits are fixed in §4.5.
- **Price source.** The oracle and averaging window used for the daily $ALL price.
- **Buyback and burn.** What counts as a platform fee, how the daily buybacks are executed and what is done with the $ALL bought back. The 50% of platform fees used for buybacks from launch and the quarterly burn of half the $ALL paid for credits are fixed in §4.6.
- **Treasury sizing.** How the Compute Treasury's reserve and staking schedule are sized against expected usage.
- **Creator rewards.** The share of credits spent on a Hub listing or published agent that is paid to its publisher, and the payout asset and schedule.

- **Hub serving costs.** How the cost of keeping community models served is recovered (§7.3).
- **Vesting.** Schedules for core contributors, the foundation and community rewards.

---

§ 05

BLIND ROUTING AND PRIVACY TIERS

5.1 THREAT MODEL

The adversaries considered are: a model provider that wishes to profile users; Allwit as an operator, including under legal compulsion; a passive network observer; and any single compromised infrastructure component. The goal is *unlinkability*: no single one of these learns both a user's identity (wallet or network address) and the content of that user's requests. Content confidentiality from the model provider is a separate goal, met only by the Private tier (§5.4). Collusion between the relay and the gateway defeats unlinkability; Phase 3 reduces that risk by attesting the gateway and diversifying relay operators (§11). Traffic analysis by a global observer, and identification from prompt content the user writes themselves, are out of scope.

5.2 THE REQUEST PATH

A request proceeds in four steps (Figure 2).

1. **Tokens.** The client already holds blind credit tokens from the issuer (§4.7).
2. **Seal.** The client encrypts the request and its tokens to the gateway's published key using Oblivious HTTP [8] and sends it to the relay. The relay sees the client's address and an opaque ciphertext.
3. **Redeem and route.** The gateway decrypts, verifies and spends the tokens, screens the request in memory (§13), and forwards it to the chosen model. For Blind-tier models it calls the provider under Allwit's own key, so the provider sees Allwit as the customer.
4. **Return.** The response is encrypted back to the client along the same path. Nothing is written to persistent storage beyond aggregate counters and the spent-token set.

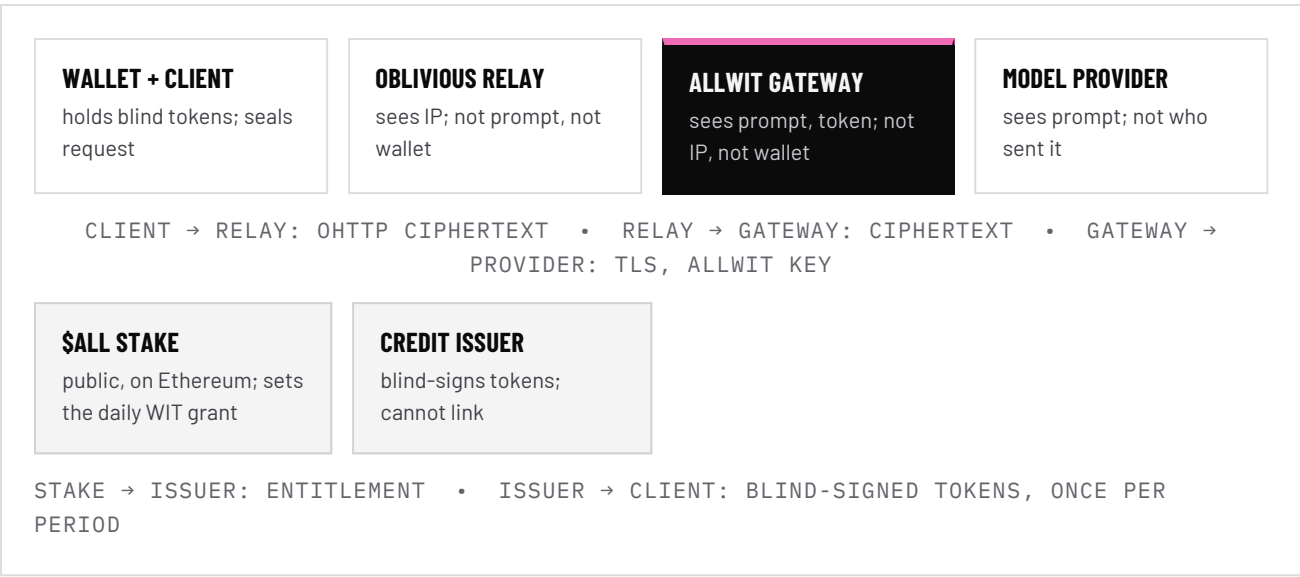

FIGURE 2: Blind routing. The wallet is used only at issuance. The relay holds the network address, the gateway holds content, and the provider holds content attributed to Allwit. No component holds both identity and content.

5.3 WHAT EACH PARTY LEARNS

PARTY	LEARNS	DOES NOT LEARN
Credit issuer	Wallet, stake, number of tokens issued per period	Which tokens were spent, on what, from where
Relay	Client network address, request timing and size	Content, tokens, wallet
Gateway	Request and response content, model, tokens redeemed	Network address, wallet
Blind-tier provider	Request and response content	Network address, wallet, any per-user identifier
Private-tier operator	Nothing in the clear when served in a TEE (Phase 3); content otherwise	Network address, wallet
Ethereum observers	Stakes, unstakes and credit purchases by address	Any usage

TABLE 4: Knowledge partition under blind routing, assuming no collusion between relay and gateway.

5.4 PRIVATE AND BLIND TIERS

Blind routing hides the requester; it does not hide content from a third-party provider. Allwit therefore labels each model by who, outside the user, can read the prompt.

- **Private.** An open, Hub or native model served on hardware Allwit controls or on decentralised compute it has qualified, with no prompt logging. From Phase 3 the Private tier runs in attested confidential environments, so the operator cannot read content either (§11.2).

- **Blind.** A third-party provider's model. The provider sees content, under its own data policy, but not the requester.

Labels are shown in the model picker, returned in API responses and enforced by policy: a user or agent may require Private and the router will refuse a Blind route rather than fall back to one (§6.3).

5.5 NO-LOG OPERATION

The gateway keeps no prompt or response logs, holds sessions only for the duration of a streamed response and performs content screening in memory. Operational telemetry is limited to aggregate counts by model and status, latency histograms and the spent-token set. In Phases 1 and 2 this is an operator commitment; Phase 3 makes it verifiable.

5.6 TOWARD AN ATTESTED GATEWAY

In Phase 3 the gateway runs inside a trusted execution environment [21], [22] whose software image is published and reproducibly built. Clients fetch the gateway's attestation before sealing requests and refuse keys not bound to a published image, following the pattern established by Private Cloud Compute [26]. The no-log property then rests on the published code and the TEE vendor's attestation chain rather than on Allwit's word (§11.3).

---

§ 06

THE MODEL GATEWAY

The gateway places many providers and open models behind one endpoint, so a user selects the best model for each task without a separate account, key or subscription per provider. It performs four functions on every request, whether the request originates in chat, the API or an agent.

6.1 ONE INTERFACE

Allwit exposes a chat application and an OpenAI-compatible API [3] for text, image, video and speech. Existing SDKs and tools work by changing the base URL and supplying a client that seals requests and attaches tokens. A thin local proxy performs sealing for tools that cannot be modified.

MODALITY	OPERATIONS	INPUTS → OUTPUTS
Text	Chat, write, code, reason, summarise	Text, files → text
Image	Generate, edit, upscale, describe	Text, image → image, text
Video	Generate short clips, animate stills	Text, image → video
Speech	Text-to-speech, speech-to-text, voice chat	Text, audio → audio, text

TABLE 5: Modalities at launch.

6.2 ROUTING AND AUTO MODE

A request names a model or asks for `auto`. In auto mode the router chooses by task, price, latency and the caller's privacy requirement. Phase 1 uses rules and published benchmarks; Phase 4 replaces them with a native router model trained to predict which model will answer a request adequately at least cost, following learned routing and cascade approaches [24], [25] (§12.1). Because the gateway keeps no logs, router training cannot use production traffic (§12.2).

6.3 FALLBACK

If a provider is unavailable or rate-limited, the router retries on an equivalent model. Equivalence classes are declared per model and never cross a privacy constraint: a request that requires Private cannot fall back to a Blind model, and a request that names an exact model fails rather than silently substituting. The response always states the model that served it.

6.4 METERING

Every model has a price in WIT credits per unit of work (per token for text, per image, per second of video or audio), derived from its cost and published in the app and the API, to two decimal places. The gateway prices a request before dispatch where possible, holds sufficient tokens, and settles on completion with a refund token for any difference (§4.7). One meter covers models, Hub deployments and agent steps. Prices are revised as provider charges change; the credit itself does not move with the $ALL price, only the size of each wallet's daily grant does (§4.4).

6.5 THE LAUNCH CATALOGUE

Launch pairs frontier models, reached through each provider's API, with open-weight models that Allwit hosts or runs on qualified decentralised compute. The families in Table 6 are candidates; exact models are fixed at launch and subject to each provider's commercial terms.

MODALITY	FRONTIER (BLIND TIER)	OPEN (PRIVATE TIER)
Text	Anthropic Claude, OpenAI GPT, Google Gemini, xAI Grok	Meta Llama, Mistral, DeepSeek, Qwen
Image	OpenAI image models, Google Imagen, Midjourney (if API access allows)	FLUX, Stable Diffusion
Video	Google Veo, OpenAI Sora, Runway	Open video models such as Wan and HunyuanVideo
Speech	ElevenLabs, OpenAI voice models	Whisper (speech-to-text), open TTS models

TABLE 6: Candidate model families at launch, by modality and privacy tier.

6.6 PROVIDER FIDELITY

An aggregator is itself trusted to send each request to the model it names. A user of Allwit cannot inspect provider keys, and blind routing removes the user's account at the provider through which they might otherwise check. Allwit therefore publishes, per model, periodic distributional tests of served outputs against reference samples, using model equality testing [17]. For Private-tier models served from a published, content-hashed artifact (§7.2), Phase 3 attestation binds the served weights to the hash. For Blind-tier models, fidelity remains a statistical audit, not a guarantee (§14).

---

§ 07

ALLWIT HUB

The Hub is a public catalogue where researchers, companies and independent builders publish open-weight models, adapters, datasets and agent templates. It serves the role of an existing model hub [2] with one difference that ties it to the rest of the platform: a served listing is not only downloadable but immediately usable through the same chat application, API, balance and labels as every other model, and an agent template opens directly in the Studio.

7.1 LISTINGS

LISTING	CONTENTS	WHAT USERS CAN DO
Models	Open-weight base and instruction models in any supported modality	Read the card, try in chat, call through the API, download where the licence allows
Adapters	Fine-tunes and low-rank adapters [13] over a base model	Run on a base Allwit already serves, at small marginal cost
Datasets	Public, licensed or opt-in data for training and evaluation	Browse, download, fine-tune a listed model
Agent templates	Agents published from the Studio	Use as published, or copy into the Studio and modify

TABLE 7: Hub listing types.

7.2 THE LISTING CARD

- **Identity.** Name, publisher wallet, version, and a content hash of the exact files [23], so that what Allwit serves can be checked against what was published.
- **Licence.** Declared at upload and enforced at serving (§7.4).
- **Capabilities.** Modality, context length, languages, evaluation results, intended use and known limitations.
- **Run details.** Privacy tier, price in WIT credits and serving state: served, on demand, or download only.

7.3 FROM LISTING TO SERVED MODEL

Publishing and serving are separate, so the catalogue can be broad while the served set stays fast.

1. **Publish.** A publisher signs in with a wallet and uploads weights or links an existing repository, with card and licence. Files are stored content-addressed on decentralised storage or mirrored from existing hubs. Only safe serialisation formats such as safetensors [16] are accepted, and every upload is scanned.
2. **Request a deployment.** The publisher or any user requests that a listing be served. Allwit checks licence, size and expected demand.
3. **Serve.** Approved models are deployed on Allwit-run or qualified decentralised compute using paged-attention serving [15] and join the Private tier. Adapters load over an already-served base model with batched multi-adapter serving [14], which is what makes long-tail fine-tunes economical to offer.
4. **Scale down.** Listings with little use move to on-demand or download-only, so popular models stay warm without paying to keep every model resident.

Who bears the cost of the served set, and how it is recovered from usage, is a financial-model question (§4.9).

7.4 LICENCE ENFORCEMENT

The serving layer reads the declared licence and enforces it: a non-commercial model is never served for paid use; gated licences must be accepted by the user's wallet, by signature, before first use; and attribution requirements are surfaced in responses. Licence declarations are the publisher's responsibility, and a takedown process handles disputes (§7.6).

7.5 CREATOR REWARDS

When a listing or published agent is used, a share of the WIT credits spent is attributed to the publisher's wallet. Attribution is computed at the gateway from aggregate per-listing counters, which reveal how often a listing is used but not by whom. The share, payout asset and schedule are `TBD` in the financial model. Rewards can be withheld from listings removed for rule violations.

7.6 COMMUNITY AND MODERATION

Publishers are identified by wallet; organisations may verify a wallet to earn a verified badge. Usage counts, ratings and open evaluations help good work surface, and each listing shows how often it is actually run. Content rules, a takedown process for licence and rights complaints, and removal of malicious or unsafe uploads apply to all listings. Browsing requires no wallet; publishing, running and earning do.

---

§ 08

AGENT STUDIO

Agent Studio lets anyone build an agent without code: an assistant, a workflow or an automation that uses models from across the catalogue, connects to tools and data, and runs under the same wallet, credits and labels as the rest of Allwit. Because the Studio sits above the whole catalogue, an agent is not bound to one vendor. It can plan with a frontier model, perform a sensitive step on a Private-tier Hub fine-tune and produce output with an image or speech model, and any of these can be replaced as better models appear.

8.1 BUILDING BLOCKS

BLOCK	WHAT IT SETS	EXAMPLES
Models	Model per step, or auto	A frontier model to plan, a Hub fine-tune for a specialist step, an image model for output
Instructions	Role, tone, rules, steps	"Summarise each new report and draft a reply in our style"
Knowledge	Sources the agent may consult	Documents, websites, a Hub dataset
Tools	Actions, each granted explicitly	Web search, code execution, media generation, external APIs, reading on-chain data
Memory	State kept between sessions	Preferences, ongoing tasks, past results
Triggers	What starts a run	A chat message, an API call, a schedule, an on-chain event

TABLE 8: The six building blocks of an agent.

8.2 EXECUTION MODEL

An agent is a declarative definition: a graph of steps, each naming a model or tool, with instructions, inputs and limits. The Studio drafts definitions from a plain-language description or a Hub template, and a playground shows each step, the model used and the credits spent before an agent is shared. At run time the definition executes in an isolated runner. Every model call the runner makes is sealed, token-paid and blind-routed exactly as a chat request is, so an agent run is indistinguishable to the gateway from a sequence of ordinary requests. Tools are exposed through a standard tool protocol [19], so third-party tool servers can be added without platform changes.

8.3 OWNER-ENCRYPTED STATE

Agents need state, which sits uneasily with a platform that keeps no logs. The Studio resolves this by holding state that Allwit stores but cannot read. An agent's knowledge, memory and configuration are encrypted under a key derived from a signature by the owner's wallet. The runner decrypts state only within a run that

the owner, or a holder of a delegated key the owner has issued, has authorised. Scheduled and event-triggered agents therefore require a delegated run key, scoped to one agent and revocable. In Phase 3, runners execute in attested environments, which extends the same guarantee to state in use.

8.4 PRIVACY LABELS COMPOSE

An agent's label is the weakest among the models and tools it may invoke. An agent that calls any Blind-tier model, or any tool that contacts an external service, is labelled Blind; one that uses only Private-tier models and no external tools is labelled Private. The Studio shows, before a run, every outside service the agent can contact. A builder can pin an agent to Private, after which the router refuses Blind routes for it (§6.3).

8.5 TOOLS, PERMISSIONS AND INJECTION

Tools are disabled unless granted, code runs in a sandbox with no network by default, and grants are scoped per tool. Agents that read untrusted content, such as web pages, email or documents, are exposed to indirect prompt injection [18]. The Studio mitigates this structurally rather than by filtering alone: tool calls with side effects can require human confirmation, content from untrusted sources is marked as data in the step graph, and spend and action limits bound what a compromised step can do (§8.6). These measures reduce, but do not eliminate, the risk (§17).

8.6 WHO PAYS, AND SPEND LIMITS

A run spends WIT credits from whoever runs it, at the price of the models and tools it calls. When a user runs an agent someone else published, the user pays and the publisher earns a creator reward (§7.5). Owners can cap credits per run and per day, so a looping or misconfigured agent cannot drain a balance.

8.7 AGENTS WITH WALLETS

In Phase 4 an agent may hold its own smart-contract wallet on Ethereum, built on account abstraction [11]. The owner sets policy in the wallet itself: allow-listed contracts, per-day spending caps, and human approval above a threshold. Policy enforced by the wallet contract holds even if the agent's reasoning is manipulated, which is the property that makes delegating funds to an agent defensible. Uses include portfolio monitors, payment-triggered workflows and agents that pay other agents for work.

---

§ 09

PHASE 1: ALLWIT GATEWAY

9.1 RATIONALE

The gateway is the smallest complete product: it delivers the core promise, the best models in one place without an identity, and it builds the foundation that every later surface requires. Starting here concentrates early engineering on identity, credits and the request path, where errors are most expensive to correct later.

9.2 SCOPE

- Frontier and open models across all four modalities (§6.5), behind the chat application and API.
- Wallet sign-in, $ALL staking on Ethereum, the daily WIT grant, credit purchases in USDC, USDT or $ALL, the daily $ALL buyback from platform fees, and blind credit tokens (§4).
- Blind routing through one independent relay operator; no-log gateway by policy (§5).
- Private and Blind labels, routing, fallback and metering (§6).
- Hub browsing and publishing may open before Phase 2, so the catalogue is stocked when serving launches.

9.3 TRUST MODEL AND LIMITATIONS

A Phase 1 user trusts that the gateway operator keeps no logs and does not collude with the relay operator; that Private-tier hosts do not read content; and that Blind-tier providers honour their own data policies. These are operational commitments. Allwit publishes them, names the relay operator, and publishes provider fidelity audits (§6.6). Blind tokens and the relay split are cryptographic and do not depend on these commitments for the issuer's inability to link spending to a wallet.

---

§ 10

PHASE 2: HUB AND AGENT STUDIO

10.1 SERVING THE HUB

Phase 2 turns on deployment requests, serving and scale-down for Hub listings (§7.3), licence enforcement (§7.4) and creator rewards (§7.5). Served listings join the model picker and the API in the Private tier.

10.2 AGENT STUDIO LAUNCH

The Studio launches with the six building blocks, the playground, owner-encrypted state, label composition, tool permissions and spend caps (§8). Agents can be kept private, shared by link, called through the API or published to the Hub.

10.3 TRUST MODEL

Phase 2 adds two assumptions to Phase 1. Users trust publishers' licence declarations, backed by takedown. And agent runners, until Phase 3, are trusted to handle decrypted state only for the duration of a run; owner encryption protects state at rest, not in use.

---

§ 11

PHASE 3: OPEN NETWORK

11.1 INDEPENDENT RELAYS

The relay role opens to several independent operators, including $ALL stakers who run relay nodes. Clients choose a relay per session. Relay operators handle only ciphertext and network addresses, so opening the role adds no content exposure, and diversifying operators makes relay-gateway collusion harder. Whether and how relay operators are compensated is a financial-model question.

11.2 PRIVATE COMPUTE TIER

Private-tier models move to confidential GPU environments [20] whose attestation binds the running image and the model's content hash. The operator of the hardware, whether Allwit or a qualified decentralised provider, can then no longer read prompts, and a client can check which weights served its request.

11.3 ATTESTED NO-LOG GATEWAY

The gateway moves into a TEE with a published, reproducibly built image (§5.6). Clients verify attestation before sealing to the gateway's key. A change to the gateway's code becomes a published, inspectable event rather than an internal deployment.

11.4 TRUST MODEL

After Phase 3, unlinkability no longer depends on Allwit's operational promise. It depends on the TEE vendors' hardware and attestation chain, on the published gateway code being correct, and on relay operators not colluding with a compromised gateway. Blind-tier providers still see content.

---

§ 12

PHASE 4: NATIVE

12.1 NATIVE ROUTER

A native router model replaces rule-based auto mode, choosing the cheapest model expected to answer adequately (§6.2). It runs inside the attested gateway, so routing decisions about a request's content are subject to the same no-log guarantee as the request itself.

12.2 NATIVE MODELS AND TRAINING DATA

Allwit's own models join the same catalogue. Their purpose is what a reseller cannot offer: Private by default, priced in fewer WIT credits than frontier equivalents because no third-party provider is paid, and tuned for the platform, for example a router, an assistant that understands wallets and on-chain data, and agent-oriented models. The practical path is to fine-tune strong open-weight bases before considering training

from scratch. Usage cannot be a training source: blind routing and no-log operation mean Allwit does not hold user prompts. Training data must come from licensed, public or explicitly opt-in sources, including Hub datasets whose licences permit it.

12.3 AGENTS ON ETHEREUM

Agents gain policy-bound wallets (§8.7) and on-chain triggers, enabling assistants that act within owner-defined limits.

12.4 TRUST MODEL

Phase 4 adds no new trust for model use. For agents with wallets, users additionally trust the policy contract's code, which is public and should be audited before release.

---

§ 13

RESPONSIBLE USE WITHOUT IDENTITIES

Not knowing who sends a request cannot mean not caring what is sent. Providers hold resellers responsible for misuse, and the Hub and Studio add uploaded content and automated actions. Allwit therefore acts on requests, tokens, listings and agents rather than on people.

- **In-memory screening.** Requests and outputs are checked against usage rules as they pass the gateway and blocked without being stored. In Phase 3 the screening code is part of the attested image.
- **Limits on tokens, not people.** Rate and spend limits apply per token batch and per wallet at issuance, so abuse is throttled without identifying anyone. A wallet's daily grant itself bounds how much it can spend in a period.
- **Provider terms.** Frontier models are offered only under commercial agreements that permit Allwit's model, and each provider's usage policy applies to requests routed to it.
- **Accountable publishing.** Hub listings and published agents are tied to a publisher wallet and can be removed, together with their pending creator rewards, for breaking the rules.

The design accepts a real cost: Allwit cannot attribute an individual abusive request to a wallet after the fact. It compensates by bounding what any wallet can do in a period and by refusing content at the gateway. Whether this satisfies each frontier provider is a commercial question that Phase 1 must settle before launch (§17).

---

§ 14

TRUST PROGRESSION AND SECURITY SUMMARY

14.1 THE TRUST ASSUMPTIONS, STATED

PROPERTY	PHASES 1–2	PHASE 3 ONWARD
Issuer cannot link spending to a wallet	Cryptographic (blind signatures)	Cryptographic
Gateway cannot see network address	Cryptographic (OHTTP), given an honest relay	Same; several relay operators
Gateway keeps no logs	Operator commitment	Attested published image
Relay and gateway do not collude	Separate operators by agreement	Attested gateway; diverse relays
Private-tier operator cannot read content	Operator commitment	Confidential computing
Served open model matches its hash	Operator commitment; audits	Attested
Blind-tier model is the one named	Statistical audit	Statistical audit
Blind-tier provider handles content per policy	Provider's policy	Provider's policy
Agent state confidential at rest	Owner encryption	Owner encryption
Agent state confidential in use	Runner commitment	Attested runner

TABLE 9: Trust assumptions by phase. Rows marked "commitment" are promises in the early phases and are the targets of Phase 3.

14.2 INDEPENDENCE OF THE LAYERS

The properties fail independently. A compromised relay learns addresses but no content. A compromised gateway learns content but neither addresses nor wallets. A compromised issuer learns stakes, which are public anyway, but not spending. Linking a person to a prompt requires compromising, or colluding across, at least two of relay, gateway and issuer, and in Phase 3 the gateway compromise must also defeat attestation. No layer's failure silently degrades another; each is separately observable by users through published operators, keys and images.

§ 15

MARKET DESIGN AND POSITIONING

Allwit competes on a combination no existing product offers: breadth of frontier and open models, a hub where community models run, agents across every model, privacy from both providers and the platform, and payment without a card. Account-based aggregators match breadth but not privacy or payment. Model hubs match openness but run only a small curated subset and rarely pay creators from usage. Single-vendor agent builders match ease of use but bind agents to one provider. Decentralised inference networks address supply rather than the user's identity.

The intended users are people and teams who use several models daily and object to tying a prompt history to their identity; developers who want one API and one balance with fallback; open-source creators who want their work run and rewarded; builders of agents that must not be locked to one vendor; and $ALL holders, for whom staking turns a token into a daily allowance of compute that grows with the token's value. The Hub and Studio also compound one another: published agents drive usage of Hub models, and Hub models give agents capabilities a single vendor would not.

---

§ 16

GOVERNANCE

At launch Allwit is operated by its founding team, which sets parameters, admits providers and Private-tier compute, and runs takedown. This paper does not specify on-chain governance. Decisions that a future governance process would naturally own include: financial-model parameters (Appendix A), the Compute Treasury's staking schedule, the use of bought-back $ALL, admission of relay operators and decentralised compute providers, content and listing rules, the creator reward share, and approval of new gateway images in Phase 3. Whether any of these move to $ALL holders, and by what mechanism, will be addressed alongside the financial model.

---

§ 17

LIMITATIONS AND OPEN PROBLEMS

- **Compute funding.** Providers bill in fiat while each wallet's WIT grant tracks the $ALL price, so the Compute Treasury must be sized against usage across a range of prices. The treasury's reserve, its staking schedule and a catalogue mix weighted toward open models are central to the financial model, which is not yet specified.
- **Frontier provider terms.** Serving anonymous users under Allwit's own provider keys requires explicit commercial agreements. Without them, a provider could revoke access.

- **Anonymity set size.** Token unlinkability is only as strong as the number of wallets issued under the same key in the same period. Early, with few users, the set is small; issuance keys must be few and rotated on a fixed schedule to avoid partitioning users.
- **Grant size as a side channel.** Because a wallet's daily grant follows its stake, the number of tokens issued could in principle narrow down which wallet a batch belongs to. Issuing in fixed denominations and batching issuance across wallets limits what the token count reveals; the residual leakage is to be quantified.
- **Timing correlation.** A wallet's issuance time, and the first requests after it, can be correlated by a party observing both issuer and gateway. Clients should obtain tokens on a schedule independent of use.
- **Content identifies.** Prompts can identify their author. Blind routing hides metadata, not what a user chooses to write.
- **Ethereum costs.** Mainnet transaction fees make small stakes and frequent stake changes expensive. Mitigations, including batched operations, are open.
- **Prompt injection.** Agents that read untrusted content remain exposed to injection [18]; structural limits reduce its impact but do not remove it.
- **Hub cost and liability.** Serving community models costs compute, and licence and rights complaints require a working takedown process.
- **TEE trust.** Attestation shifts trust to hardware vendors and their supply chains, and TEEs have a history of side-channel vulnerabilities.
- **Scope.** Three surfaces is a large build. Phasing keeps Phase 1 small enough to ship.

---

§ 18

CONCLUSION

Allwit brings together what today exists only separately: broad access to frontier and open models, a hub where community models can be run, a studio for building agents across all of them, privacy from both providers and the platform, and a way to pay without a card or an identity. It does so by composing established primitives, wallet sign-in, blind-signed tokens, oblivious transport and confidential computing, so that identity and content never meet in one place, and by labelling plainly where a third party must still see content. Access is priced by stake rather than by identity: a wallet that holds $ALL receives WIT credits every day in proportion to what it has staked and what that stake is worth, and the Allwit Compute Treasury pays for the compute those credits buy. Usage is tied to the token: from launch, half of platform fees buy $ALL back every day, and half of the $ALL paid for credits is burned each quarter. Phase 1 delivers a complete gateway; Phase 2 turns Allwit from a place to use AI into a place to build and share it; Phase 3 replaces operational promises with attested mechanisms; Phase 4 adds native models and agents that act within enforced limits. The financial model, which this paper leaves open, will set the remaining parameters: the tiers, the credit packages, and how creators and relay operators share in usage.

APPENDIX A

PARAMETER REFERENCE

Parameters referenced in this paper, with the values this draft fixes and those that belong to the financial model.

PARAMETER	WHERE USED	VALUE
$ALL total supply	§4.2, §4.3	1,000,000,000, fixed
$ALL distribution	§4.3	Table 2
Public auction	§4.3	90 days, pre-TGE; 400,000,000 $ALL in daily allocations of about 4.44M
Auction price	§4.3	$0.05 floor on day 1; +1% the next day if 50% or more of the day's tokens sell, otherwise unchanged
Unsold auction tokens	§4.3	Carried to the next day; after day 90 burned or sent to the Compute Treasury, TBD
Uniswap pool opening price	§4.3	Final auction clearing price
Auction proceeds to the Compute Treasury	§4.3	50%, held in stablecoins
Compute Treasury staking schedule	§4.3	About a quarter of its allocation in year 1; remainder TBD
Daily WIT grant per wallet	§4.4	Function of $ALL staked and $ALL price
$ALL price source and averaging window	§4.4	On-chain oracle, time-weighted average; window TBD
Grant reset period	§4.4	24 hours
Tier thresholds (entry-level to team tier)	§4.5	TBD
Unstaking cooldown	§4.5	TBD
Retail price of a WIT credit	§4.2, §4.5	$0.01: 100 credits per USDC or USDT; $ALL converted at its oracle price at purchase
Validity of credits	§4.4, §4.5	Staking credits expire at each reset; purchased credits carry over
Balance precision	§4.2	Two decimal places
Credit package sizes	§4.5	TBD
Split of credit purchases	§4.6	USDC, USDT: to providers, infrastructure and development. $ALL: 50% to the Compute Treasury; 50% burned
Daily $ALL buyback	§4.6	50% of platform fees, every day from launch; use of the $ALL bought back TBD

PARAMETER	WHERE USED	VALUE
Burn of $ALL paid for credits	§4.6	50% of the $ALL paid for credits, at the end of each quarter
Token denominations	§4.7	TBD
Issuance key rotation period	§4.7, §17	TBD
Per-model credit prices	§6.4	Set from cost; published in app and API
Creator reward share	§7.5	TBD
Relay operator compensation	§11.1	TBD
Default agent spend caps	§8.6	TBD
Vesting schedules	§4.3	TBD

TABLE 10: Parameter reference. Values marked TBD will be supplied by the companion financial model.

APPENDIX B

ANATOMY OF A SEALED REQUEST

An illustrative, decrypted view of a request as the gateway sees it after removing the OHTTP encapsulation. Field names are not final. Note what is absent: no wallet, no session, no network address.

```
// inner request, encrypted to the gateway key and sent via the relay
{
  "model": "auto",
  "privacy": "private",            // refuse Blind-tier routes
  "max_credits": 4.00,
  "tokens": [                      // blind-signed, unlinkable to issuance
    { "key_id": "iss-2026-10-27-a", "denom": 2.00, "nonce": "b64…", "sig": "b64…" },
    { "key_id": "iss-2026-10-27-a", "denom": 2.00, "nonce": "b64…", "sig": "b64…" }
  ],
  "request": {                     // OpenAI-compatible body
    "messages": [ { "role": "user", "content": "…" } ],
    "stream": true
  }
}

// response metadata returned to the client
{
  "served_by": "qwen3-32b@Allwit-private",
  "tier": "private",
  "credits_charged": 3.10,
  "refund": { "denom": 0.90, "sig": "b64…" }  // blind refund token
}
```

LISTING 1: Illustrative sealed request and response metadata. All amounts are in WIT credits, to two decimal places.

APPENDIX C

OPEN QUESTIONS AND FUTURE WORK

C.1 FINANCIAL MODEL

All items in §4.9 and Appendix A, including tier thresholds, credit package sizes, treasury sizing, the use of bought-back $ALL and the creator reward share.

C.2 ISSUANCE PRIVACY AT SMALL SCALE

Minimum anonymity set sizes, key rotation schedules and client issuance timing that keep unlinkability meaningful while the user base is small, and the residual leakage from grant sizes (§17).

C.3 ABUSE CONTROLS UNDER ANONYMITY

Whether per-wallet issuance limits and in-memory screening satisfy each frontier provider's usage policy, and what evidence providers will require.

C.4 ATTESTATION AND REPRODUCIBLE BUILDS

The build pipeline, transparency log and client verification flow for the Phase 3 gateway and Private-tier images.

C.5 QUALIFYING DECENTRALISED COMPUTE

Criteria under which third-party compute may serve the Private tier, and how attestation and model hashes are checked for it.

C.6 AGENT SAFETY

Default confirmation policies for side-effecting tools, and formal limits for agent wallet policy contracts.

C.7 GOVERNANCE

Which decisions, if any, move to $ALL holders, and the mechanism for doing so.

SOURCES

REFERENCES

[1] OpenRouter. A unified interface for LLMs. openrouter.ai

[2] Hugging Face. The Hugging Face Hub. huggingface.co/docs/hub

[3] OpenAI. API reference: Chat Completions. platform.openai.com/docs/api-reference

[4] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of Crypto 82*, 1983.

[5] A. Davidson, J. Iyengar, C. A. Wood. The Privacy Pass Architecture. RFC 9576, IETF, 2024.

[6] S. Celi, A. Davidson, S. Valdez, C. A. Wood. Privacy Pass Issuance Protocols. RFC 9578, IETF, 2024.

[7] F. Denis, F. Jacobs, C. A. Wood. RSA Blind Signatures. RFC 9474, IETF, 2023.

[8] M. Thomson, C. A. Wood. Oblivious HTTP. RFC 9458, IETF, 2024.

[9] W. Chang, G. Rocco, B. Millegan, N. Johnson, O. Terbu. EIP-4361: Sign-In with Ethereum. Ethereum Improvement Proposals, 2021.

[10] F. Vogelsteller, V. Buterin. EIP-20: Token Standard. Ethereum Improvement Proposals, 2015.

[11] V. Buterin et al. ERC-4337: Account Abstraction Using Alt Mempool. Ethereum Improvement Proposals, 2021.

[12] WalletConnect. WalletConnect specifications. specs.walletconnect.com

[13] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen. LoRA: Low-Rank Adaptation of Large Language Models. ICLR, 2022. arXiv:2106.09685.

[14] Y. Sheng et al. S-LoRA: Serving Thousands of Concurrent LoRA Adapters. MLSys, 2024. arXiv:2311.03285.

[15] W. Kwon et al. Efficient Memory Management for Large Language Model Serving with PagedAttention. SOSP, 2023.

[16] Hugging Face. safetensors. github.com/huggingface/safetensors

[17] I. Gao, P. Liang, C. Guestrin. Model Equality Testing: Which Model Is This API Serving? ICLR, 2025. arXiv:2410.20247.

[18] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, M. Fritz. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. AISec, 2023. arXiv:2302.12173.

[19] Model Context Protocol. Specification. modelcontextprotocol.io

[20] NVIDIA. Confidential Computing on NVIDIA Hopper GPUs. nvidia.com

[21] Intel. Intel Trust Domain Extensions (Intel TDX). intel.com

[22] AMD. AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More. White paper, 2020.

[23] J. Benet. IPFS: Content Addressed, Versioned, P2P File System. arXiv:1407.3561, 2014.

[24] I. Ong, A. Almahairi, V. Wu, W.-L. Chiang, T. Wu, J. E. Gonzalez, M. W. Kadous, I. Stoica. RouteLLM: Learning to Route LLMs with Preference Data. ICLR, 2025. arXiv:2406.18665.

[25] L. Chen, M. Zaharia, J. Zou. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. arXiv:2305.05176, 2023.

[26] Apple Security Engineering and Architecture. Private Cloud Compute: A new frontier for AI privacy in the cloud. Apple Security Research, 2024. security.apple.com

[27] Y. Rao, J. Steeves, A. Shaabana, D. Attevelt, M. McAteer. BitTensor: A Peer-to-Peer Intelligence Market. arXiv:2003.03917, 2020.

ALLWIT

INTELLIGENCE FOR ALL

YOUR KEYS.
YOUR PRIVACY.

Use, build and share AI without trading away who you are.