AdEx

The Solution For **Digital Advertising** That **Improves Transparency** And **Protects User Privacy.**

WHITEPAPER

AdEx Protocol

Intro

AdEx is an open, trust-minimized protocol & stack for digital advertising that reduces ad fraud, malvertising and protects use privacy. It originated in 2017 as a decentralized ad exchange for digital advertising, and later evolved into the full-stack solut is today.

The AdEx protocol facilitates trading of advertising space/time, as well as the subsequent verification and proof that it actua occurred. Essentially, it covers all interactions between publishers, advertisers and end users. The protocol combines traditior peer-to-peer technology, cryptography and blockchain.

The rationale for creating the AdEx protocol is to create an open-source, transparent and fraud-proof replacement to the ex stack. In a way, AdEx's mission is to create a new standard in digital advertising: by introducing real-time tracking and report directly accessible to each advertiser and publisher, and dropping the need for most intermediaries, we dramatically reduce ability for any side to report wrong data to others for their own financial gain. For more information about our rationale, see Benefits overview.

The AdEx team also develops an open source platform built on top of the Ethereum implementation of the protocol, availab https://platform.adex.network (GitHub Repository).

The AdEx protocol is designed to be completely invisible to end users, while improving their internet browsing experience (generally encouraging quality ads and unobtrusive experience).

This document assumes basic familiarity with computer science, blockchain and adtech.

Values

- **Transparency:** full reporting transparency for publishers and advertisers; they both receive the same reports
- **No intermediaries:** connect publishers and advertisers as directly as possible, therefore maximizing results and revenu this also implies minimized fees and no commissions
- **Privacy:** ensure the user's data stays private by never collecting it
- **No custody of funds:** users have full control over their own funds; you can withdraw any amount at any point, no thre
- **Censorship resistance:** we empower a free and self-governed ad market with no centralised restrictions on what can a can't be advertised
- **Ease of use:** modern adtech is complicated. We aim to make it as easy as possible to interact with AdEx
- **Flexibility:** wide variety of use cases, including but not limited to: display advertising, affiliate networks, and even cont micropayments

Terminology

Supply side

Throughout these documents, "supply side", "publisher" or "publishers" all refer to entities who sell ad inventory.

Demand side

Throughout these documents, "demand side", "advertiser", "advertisers" or "buyers" all refer to entities who buy ad inventor

Users

When we refer to "users", we mean end users: not of AdEx itself, but of the publishers. In other words, the users who see the but might not even be aware of AdEx's existence.

Events

Events, in the context of the AdView or the off-chain event aggregation, mean anything that a user does in regard to a digita for example, click, impression, closing of the web page, etc. Events are usually broadcast as signed messages.

Custom events

Custom events usually refer to events that are publisher-defined. For example, if you're a publisher with an e-commerce web you might choose to send an event for product purchases.

A potential use case is using AdEx for affiliate networks, where publishers get a share of the revenue on every purchase of a product.

Campaigns

Ad campaigns are traditionally defined as "coordinated series of linked advertisements with a single idea or theme". In AdEx further represent a intent to spend a certain budget towards spreading those advertisements: essentially, a big piece of dem

Campaigns are created with a total budget (e.g. 5000 DAI) and a specification of the desired result: e.g. purchase as many impressions as possible for this ad, with a maximum allowed price per impression and targeting information.

Because campaigns represent a financial commitment on a smart contract, they can be also be seen as smart, automated ins orders.

The cryptocurrencies that can be used for a campaign depend on what Core is used and what it supports: e.g. the Ethereum implementation supports all ERC20 tokens.

In the AdEx protocol, one campaign always maps to one payment channel called **OUTPACE**.

Layer 2

Layer 2 refers to blockchain scaling solutions, which allow financial transactions or other state transitions to happen very fast the blockchain, while still being enforceable or eventually being committed to the underlying blockchain.

Ideally, layer 2 solutions allow throughput levels seen in centralized systems, while still being as trustless and censorship-res as blockchains.

In AdEx, we use two scaling primitives that we defined: **OCEAN** and **OUTPACE**.

Off-chain event aggregation (OCEAN)

OCEAN stands for **O**ff-**c**hain **e**vent **a**ggregatio**n**.

An OCEAN channel is defined on-chain with a validator set, timeout and a definition of what we're looking to get achieved o chain. Then, the validators observe off-chain events, and the leading validator (`validators[0]`) would propose new states, an rest of the validators may check and sign those new states.

If a state is signed by a supermajority (>=2/3) of validators, it can be used to enforce a result on-chain.

Ocean-based unidirectional trust-less payment channel (OUTPACE)

OUTPACE stands for **O**cean-based **u**nidirectional **t**rustless **pa**yment **c**hann**e**l

This is a concept that builds on **OCEAN**, where each channel also has a deposit and a `validUntil` date, and each state repres tree of balances.

State transition rules

The state transition function enforces a few simple rules for each next state: (1) the sum of all balances in the state can only increase, (2) each individual balance can only increase and (3) the total sum of the balances can never exceed the channel de

One advertising campaign is mapped to a single OUTPACE channel, where the deposit is the entire campaign budget, and th validators are normally an advertiser-side and a publisher-side validators. That allows the advertiser to make micropayments multiple publishers (one micropayment per impression/click/etc.), and the publishers are able to withdraw their earnings at a point.

States

The possible states of an OUTPACE channel are:

- `Unknown`: the channel does not exist yet
- `Active`: the channel exists, has a deposit, and it's within the valid period
- `Expired`: the channel exists, but it's no longer valid
- `Exhausted`: this is a meta-state that's not reflected on-chain; it means the channel is Active, but all funds in it are spent

For a full explanation, see OUTPACE.

Validators

Validators are responsible for tracking ad impressions/clicks and signing the state. The validator set (can also be called a committee) is individually defined by each **OUTPACE** channel. Since each ad campaign is an OUTPACE channel, it has it's ow validator set.

Each validator must have a keypair and a publicly accessible HTTPS endpoint for receiving events from the AdView.

Observers

The observers are delegated to collect events in relation to a certain campaign. All validators of a campaign (OUTPACE chan are, by definition, observers of all events related to it.

However, in practice, it's possible to have additional observers who are not validators - for example, a publisher's node migh observe all events related to the ad units of the publisher, without necessarily being validators.

Each observer must have a publicly accessible HTTPS endpoint for receiving events from the AdView.

Validator stack

"Validator stack" refers to the entire validator stack, which is a set of software components that all validators/observers need run.

To prevent confusion with the normal terms "supply-side platform" (SSP) and "demand-side platform" (DSP), we will use "publisher-side validator" and "advertiser-side validator".

Flow

The entire flow is as follows:

1. The advertiser (demand side) starts a campaign with a total budget and certain parameters (ad units, targeting, min/m price per impression/click/etc.); this translates to opening an OUTPACE channel; at this point the advertiser delegates t validators: one that represents them (advertiser-side validator), and one that represents publishers (publisher-side vali
2. Validator(s) have to accept that they're nominated for this channel (and prove that they're available) by broadcasting a signed message to the other validator(s).
3. Publishers will query their own validator(s) for available demand (active channels) every time someone opens their website/app; the query will happen on the client side (in the browser/app), much like header bidding; the AdEx AdViev select one of those bids and relay that selection to the validators.
4. The AdView will generate events (impressions, clicks, page closed, etc.) and send them to the validators.
5. The events will be reflected by the validators, creating a new state; each valid impression event is turned into a micropayment to a publisher; publishers will be immediately able to use that state to withdraw their earnings.
6. Should the publisher decide to withdraw their earnings, they can withdraw from any number of channels at once.
7. As long as the state keeps advancing, publishers have a constant guarantee of their revenue; should the state stop adv properly, publishers can immediately stop serving ads (see campaign health and Campaign health vs refusal to sign).

The benefits of this approach are:

- Scalability: the only on-chain transactions are a deposit operation (which creates a campaign and a channel, `channelOp` and a withdraw (allowing any party to withdraw earnings, `channelWithdraw`);
- Publishers can withdraw their latest earnings on-chain at any time;
- Since **OUTPACE** is one-to-many, a campaign can be executed by multiple publishers;
- If new states are no longer created (someone is no longer online or is malicious), publishers can immediately stop deliv ads for this campaign (channel);
- Allows off-chain negotiations: advertisers can bid for impressions in real time;
- All data other than payments information is kept off-chain.

Each campaign has a duration, normally in the range of 2-12 weeks. An OUTPACE channel should have 2-3 times as long of a duration, in order to allow extra time for publishers to withdraw their revenue.

Closing a campaign

If an advertiser wants to close a campaign, they sign a new state, which distributes the remaining deposit: most of it goes ba the advertiser's wallet, and a small part goes to the publisher validator as a cancellation fee.

The publisher-side validator recognizes this as an intention to close the campaign, and signs the state as well, therefore allov the advertiser to withdraw their unspent funds. With this, the channel is considered exhausted and no longer represents any demand.

While it is possible for a publisher-side validator to refuse to approve the state, they gain nothing from doing so: (1) the adv has decided to cancel the campaign, meaning they won't sign any new states with new payments to publishers anyway; (2) a channel is no longer valid, they still get their unspent deposit back; and (3) the publisher-side validator gets compensated w cancellation fee.

Campaign health

The campaign health is a publisher-specific concept that indicates whether the advertiser is properly paying out after impres events.

Each publisher, with the help of the publisher-side validator, tracks the health status of each campaign they've ever interacte If a certain (configurable) threshold of non-paid impression events is reached, the campaign will be marked unhealthy, and t publisher will no longer pick it until the paid amount increases sufficiently.

The campaign health should not be confused with OUTPACE state sanity: even if a campaign is unhealthy, the publisher-side validator will continue signing new states as long as they're valid: because of the unidirectional flow, valid states can only me more revenue for publishers.

Campaign health vs Refusal to sign

While a campaign can be unhealthy and the publisher-side validator (Follower) will continue to sign new states, there are 2 c in which it will refuse to sign states:

1) In case of the **campaign health** dropping below an unsignalbe (configurable) threshold, the publisher-side validator (Foll will stop approving states proposed by the advertiser-side validator (Leader) until this threshold is surpassed again.

NOTE: The `Unsignable` and `Unhealthy` thresholds and 2 different configurable values and the latter is greater.

For example:

- `Unhealthy`: when health < 95%;
- `Unsignable`: when health < 75%;

2) **Refusal to sign on rules violation** will happen when the proposed state of the advertiser-side validator (Leader) violates the 3 rules of state transition (see also OUTPACE State transitions rules & OUTPACE.md Specification).

Validator fees

Running the validator stack requires computational resource, and the way the validator consensus works implies that channe validators have to represent opposite sides (if they don't, the channel should not be used).

This means that in most cases, no matter if you're a publisher or an advertiser, you'd end up using a validator ran by someor

Third-party validators may require fees to participate in your channel (campaign). With OUTPACE, there's a convenient way c doing that, by just including an entry in the balances tree. Furthermore, the fees can be ongoing (e.g. per 1k events, or per m taking advantage of the micropayments capability of OUTPACE.

In practice, a validator fee paid out proportionally to the distributed funds also works as the cancellation fee: if the cancel th campaign early, the full validator fee will be distributed without any real work done, giving the publisher-side validator an in to allow this.

Validator consensus

In a minimal setup, we have two validators defending opposite interests (advertiser-side, publisher-side).

This setup, by itself, does not imply any additional trust: each new state has to be approved by both the paying side and the receiving side (essentially, a 2 out of 2 setup). Essentially, the sender signs a new state, which pays more to the receiver, but require both to sign off, otherwise the sender would be able to arbitrarily manipulate the balances. To learn more, you can re Understanding payment channels or state channels.

However, in OUTPACE, unlike in regular state/payment channels, we separate participants from signing parties (validators), a allow any arbitrary number of validators.

Trust implications

For a state to be valid, it requires >=2/3 validator signatures. In a setup with the minimum number of validators, 2, this can c
mean 2 signatures.

As you many have noticed, we imply that multiple publishers delegate/operate a single publisher-side validator, implying it v
in their interest.

Generally, even without trusting the validator, the publishers will receive constant guarantees for their revenue.

However, if the publisher-side validator and the advertiser-side validator both become malicious, they can sign a new state,
allowing them to withdraw the channel balance together.

This attack is only possible if >2/3 (in this case, all 2 out of 2) validators become malicious, and it wouldn't be a problem in a
regular payment channel where the signers are the actual participants.

There are a number of mitigations that we believe are sufficient:

1. The publisher-side validator(s) should be operated by consortiums of the largest publishers;
2. There could be more than 2 validators (this also solves natural discrepancies and liveness issues);
3. Generally, there's little incentive for an advertiser-side validator to help a publisher-side validator steal a portion of *the* deposit
4. Anyone can run publisher-side validators, so we expect different publishers grouping together to create multiple valida
in other words, large publishers can run their own publisher-side validators.

Liveness implications

It's absolutely essential that validators stay online all of the time. If more than a third of them go offline, no new states can b
produced (threshold for a valid state is >=2/3 signatures), meaning that the micropayments from the advertiser to the publi
are essentially stopped.

If this happens, the publishers can immediately stop delivering ads for the given campaign mapped to the stalled channel,
therefore not losing anything. The market component is responsible for monitoring the state of all channels, and keeping tra
which ones are active and non-exhausted.

Should the validator(s) come online again, everything can resume as normal.

The possibility of validators going offline is mitigated by (1) the architecture of the validator stack and (2) the ability of OUTF
to work with any arbitrary number of validators.

Components

Core

The AdEx protocol builds on top of blockchain technology to facilitate the parts that need achieving consensus in a trust-les decentralized manner. This part is commonly referred as the "AdEx Core".

The Core has to implement everything related to moving funds between advertisers and publishers. To be more precise, it pr an implementation of OUTPACE channels (unidirectional payment channel), and every advertiser's campaign maps to one OUTPACE channel with a certain deposit.

The channel is created with the following information:

- `deposit`: total monetary deposit; on Ethereum, this is denoted in `tokenAddr` and `tokenAmount`;
- `validUntil`: the date until this channel is valid; this is also the period within the publishers can withdraw, so it should b longer than the actual specified campaign length (e.g. 3x longer);
- `validators`: an array of all the validators who are responsible for signing a new state; one of them should represent the advertiser, and the other - the publisher(s);
- `spec`: describes all the campaign criteria: e.g. buy as many impressions as possible, the maximum price they're willing t for impressions, and campaign duration; this is stored as arbitrary bytes (32); in the platform, we encode the criteria dir in there, but it can be used to reference a JSON descriptor stored on IPFS.

The Ethereum implementation of this component is called `adex-protocol-eth`. While the current running implementation of is the Ethereum one, we are also experimenting with Cosmos and Polkadot.

The on-chain interactions are:

- `channelOpen(channel)`: open an OUTPACE channel;
- `channelWithdraw(state, signatures, merkleProof, amount)`: allows anyone who earned from this channel to withdraw earnings by providing `(state, signatures)` and `merkleProof`;
- `channelExpiredWithdraw(channel)`: allows the channel creator to withdraw the remaining deposit in a channel after it e not needed on blockchain platforms where we can define our own "end block" function, like Cosmos/Polkadot.

For more information on how the payment channels work, see OUTPACE.

Market

The market is a RESTful service maintained and hosted by AdEx Network OÜ.

The primary role of the market is to facilitate demand/supply discovery and trading. It keeps record of all campaigns that are currently valid, and allows publishers/advertisers to query that list in order to find what they need.

The market needs to track all on-chain OUTPACE channels and needs to constantly monitor their liveness (>=2/3 validators and producing new states) and state.

Additional privacy can be achieved by having groups of publishers/advertisers run their own instances of the market - that w campaigns will be visible only within their private group.

The market is currently implemented in the `adex-market` repository. Because of it's aggregation-only role, it can be considere back-end to the Platform.

For a detailed specification, see market.md.

Validator stack

The validator stack is a collective term for all off-chain components responsible of handling events, managing OUTPACE cha and generating analytical reports.

Full list of functionalities:

1. Collecting events from users; this includes filtering them to ensure their validity and applying `campaignSpec` policies (e.g 10 events per user);
2. Track the on-chain state of OUTPACE channels;
3. Serve as a validator of the OUTPACE channels;
4. Generating analytical reports;
5. Providing RESTful APIs for access to reports, events and OUTPACE channel data.

In a normal setup, each of the nominated validators for an OUTPACE channel would run a full validator stack setup.

The validators communicate with the outside world and between each other through a RESTful API, exposed by a componer called a Sentry.

For a detailed specification, see validator-stack.md.

campaignSpec

In the Core, each OUTPACE channel has it's own `spec`, which is an arbitrary blob of bytes designed to contain any additional information for this channel.

In the AdEx Protocol, we use that field for a specification of the advertising campaign, by referencing a JSON blob of the `campaignSpec` format.

To do that, we set the `spec` value to a 32-byte IPFS hash of the JSON blob, using the SHA2-256 digest function.

If you're a dApp builder, it is recommended that you pin this file on your own IPFS nodes. However, this file will also be permanently stored by the Market when it's uploaded to it.

For the JSON blob specification, see `campaignSpec.md`. It includes detailed description of the campaign, including min/max impression prices, targeting, ad units and etc.; currently, the format is specific to AdEx, but AdCOM might be incorporated in future.

Paying by impression (CPM) or by click (CPC)

It's possible to pay for advertising in any way by configuring a campaign goal - e.g. by impression, by click, or even by numb user registrations (CPA).

However, the default option is always impressions as we believe that this creates the best incentives. Paying by click implies risk and unpredictability, since the publishers will be pushing impressions out without prior knowledge of how much a certa will convert.

Ultimately, the raw resource the publisher provides is impressions, and the conversion rate of the ad depends mostly on the advertiser.

Analytical reports

The validators of an OUTPACE channel are usually two instances of the validator stack: one represents the advertiser, and the represents multiple publishers.

This means they receive all the data related to this OUTPACE channel, therefore allowing them to aggregate it into useful rep

This architecture ensures that both parties get their analytical reports by aggregating the data directly from the users, which ensures reporting transparency.

Alternative implementations

The validator stack is, like anything else in the AdEx protocol, modular and replaceable.

Alternative validator stack implementations can be created, and can be useful for optimizing for particular flows/workloads.

In order to maintain compatibility with the existing AdEx infrastructure (the Platform and the AdView), you don't need to foll architecture outlined in validator-stack.md, but you need to implement the same RESTful APIs.

AdView

The primary implementation is `adex-adview-manager`, which is designed for the web.

It's important to note that the AdView is entirely browser-agnostic. It can run as a library (alongside React or any other mod framework) or in an `<iframe>` on the publisher's webpage.

There are currently no native mobile implementations, but the AdView can be easily wrapped into a `WebView` on iOS/Android will work as expected, at a small performance cost.

The AdView is responsible for:

1. Pulling all possible demand (campaigns, bids) from the Market;
2. Picking which ad to show depending on the user: this depends on a combination of price and targeting (header biddin contextual targeting);
3. Generating events (impressions as per IAB's guidelines, clicks) and sending them to all validators and observers of the ad;

Later on, if needed, it will also be responsible for:

1. Creating a cryptographic identity (keypair) for the user, if they don't already have one, and persisting it in their browse

Contextual targeting

Notice a common pattern here: **sensitive information never leaves the user's browser**, and this is achieved by shifting the process of targeting (selecting ads) to the browser itself. To achieve this, we use contextual targeting.

This works by relying on publishers to feed what they know about the context (e.g. "this page is about bicycles") and potenti the user (e.g. "this user is female") directly into the AdView API. The incentive for this is built-in: better targeted ads mean hi revenues.

This system is based on tags, which are not specified in the AdEx protocol itself and are entirely defined by network particip They could describe anything - interests, demographics, geographics and etc.

Behavioral targeting

Because contextual targeting has certain limitations (e.g. no remarketing), there is a possibility to introduce behavioral targe using `localStorage` to remember tags for the user. This will not compromise privacy, because the data collected `localStorag` not exposed to any third parties.

To achieve this, the AdView always has to be loaded from the same domain (e.g. `adex.network`), in order to ensure it always reads/writes to the same `localStorage`. This can be trust-minimized in the future through ENS, IPFS or even just using check based integrity checks.

Advertisers may report tags that allow for remarketing, such as a tag indicating that a user visited their website, or even a ta indicating they've visited a particular page, allowing for dynamic remarketing.

Blacklisting ads

Users can blacklist ads, very similarly to how ads on Google/Facebook have a cross icon on the top right corner. Once you d it will be saved locally so this ad will never be shown to you, but also reported to all publisher-side validators the AdView is a of.

While a publisher-side validators may choose to ignore such an event, it's mostly in the interest of publishers to keep track c most blacklisted ads and possibly stop serving them altogether.

An additional improvement on the AdView would be to allow users to gossip blacklists directly between each other, therefor eliminating the ability of publisher-side validators to act together and ignore blacklist events. This feature is not trivial, as it requires a reliable sybil resistance mechanism.

Security

The keypair is saved in `localStorage`. However it never holds any funds, it merely serves to identify users anonymously.

In case `localStorage` is deleted, the user will receive a new keypair and the system will start learning about them again - whi actually intended behavior (e.g. using incognito mode in the browser).

The AdEx Lounge

The AdEx Lounge (called "AdEx Profile" in the original whitepaper) is a user-facing part of AdEx that allows the user to contr ad preferences.

In particular, users can opt out of seeings certain kinds of ads.

With OUTPACE channels, it's possible for users to earn monetary rewards as well, so at some point the Lounge may be used allow for users to withdraw their funds.

There's no public implementation of the Lounge yet.

Identity

The Identity layer is currently specific to our Ethereum implementation and designed to streamline the user experience of th Platform.

It is a smart contract that allows the users of the Platform (publishers/advertisers) to:

- Use many devices (e.g. PC, mobile, HW wallet) as one identity, without having to share the same private key between t (essentially a multisig)
- Interact with the Ethereum network without needing to have ETH: fees can be paid in DAI or another ERC20 token
- Allow certain actions to be scheduled/performed automatically without needing them to be online, for example withdr funds from OUTPACE channels (called "sweeping" to distinguish it from actual withdrawing)

This solves many UX hurdles that are typical for blockchain-related applications.

In the Platform, we also allow the so-called "Quick accounts": you can sign up with an email/passphrase, and the Platform w generate and store a keypair for you, encrypted with your passphrase. Because this is suboptiomal for security, those accoun be limited (by the Market) in terms of DAI they can earn. However, thanks to the Identity layer, those accounts can be easily secured by de-authorizing the temporary keypair and authorizing a proper wallet such as Metamask/Trezor.

Some of these concepts are sometimes refered to as "smart wallets", "meta tx" or "gas abstractions".

The Identity component is implemented in the adex-protocol-eth repository.

Pre-approved tokens

While OUTPACE can work with any Ethereum token that implements the ERC20 standard, not all of them are suitable for usir campaign deposit. Some tokens have fatal bugs, others allow arbitrary minting, and some are simply not liquid enough.

This is why we came up with a set of pre-approved tokens. For now, we've decided on DAI and ADX, but we can easily allow

It's important to note that **this is not enforced on a blockchain/smart contract level**, but it's merely a UI limitation. If you that a certain token should be added, you can submit a PR to `adex-platform`.

Sign-up process

We intend to allow publishers/advertisers to sign-up to the platform using any pre-approved token (e.g. DAI, ADX), or with E leveraging Uniswap to automatically convert to one of the pre-approved tokens.

If there's a suitable way to do it, we intend to allow opening a campaign with USD/EUR by integrating the platform with a th party service that allows purchasing DAI with USD/EUR.

We are also exploring the possibilities of allowing signing up with BTC, by using HTLC-based atomic swaps or Bitcoin SPVs to exchange it for a pre-approved token.

Staking (Registry)

The Registry is an autonomous system designed to provide a list of publically accessible validators that you can nominate fo campaign.

The ultimate goal of the Registry is provide exposure for everyone who wants to be a public validator, and also to hold these validators accountable if they misbehave.

This is accomplished by having each validator who wants to be on the Registry stake ADX tokens. Every time they misbehave small portion of those tokens will be burned (slashed). This makes validators with higher stake more trustworthy, as they hav more skin in the game. The reason ADX is the only token allowed for staking is that ideally, staking for the registry would be token's primary use case, as this implies a large part of the token supply would be staked and locked up, therefore making it expensive to perform a Sybil attack.

This system differs from token curated registries in that there is no approval/rejection game, and anyone with a sufficient mi stake can be registered. Furthermore, there are specific conditions which will punish misbehavior, related to the particular mechanics of OUTPACE and the validator stack.

Because challenges may require verifying validator `NewState` and `ApproveState` messages on-chain, the Registry needs high transaction throughput. Therefore, we have decided to build it as a Substrate chain, and possibly make it part of the Polkadc network.

Nomination and staking

If a validator chooses so, they may allow users to "nominate" them: stake tokens in the validator's name (delegate), therefore receiving a pro rata share of their fee earnings, but also inheriting their slash risk. We call this "staking pools". This is the case the current staking portal. A validator may choose to run a pool and distribute their earnings in order to increase the ADX st against their name.

If you're interested in staking ADX as a token holder, you can learn more here.

Incentivized staking

As of 2021, there's an incentivized staking campaign running which generates over 50% annual percentage yield.

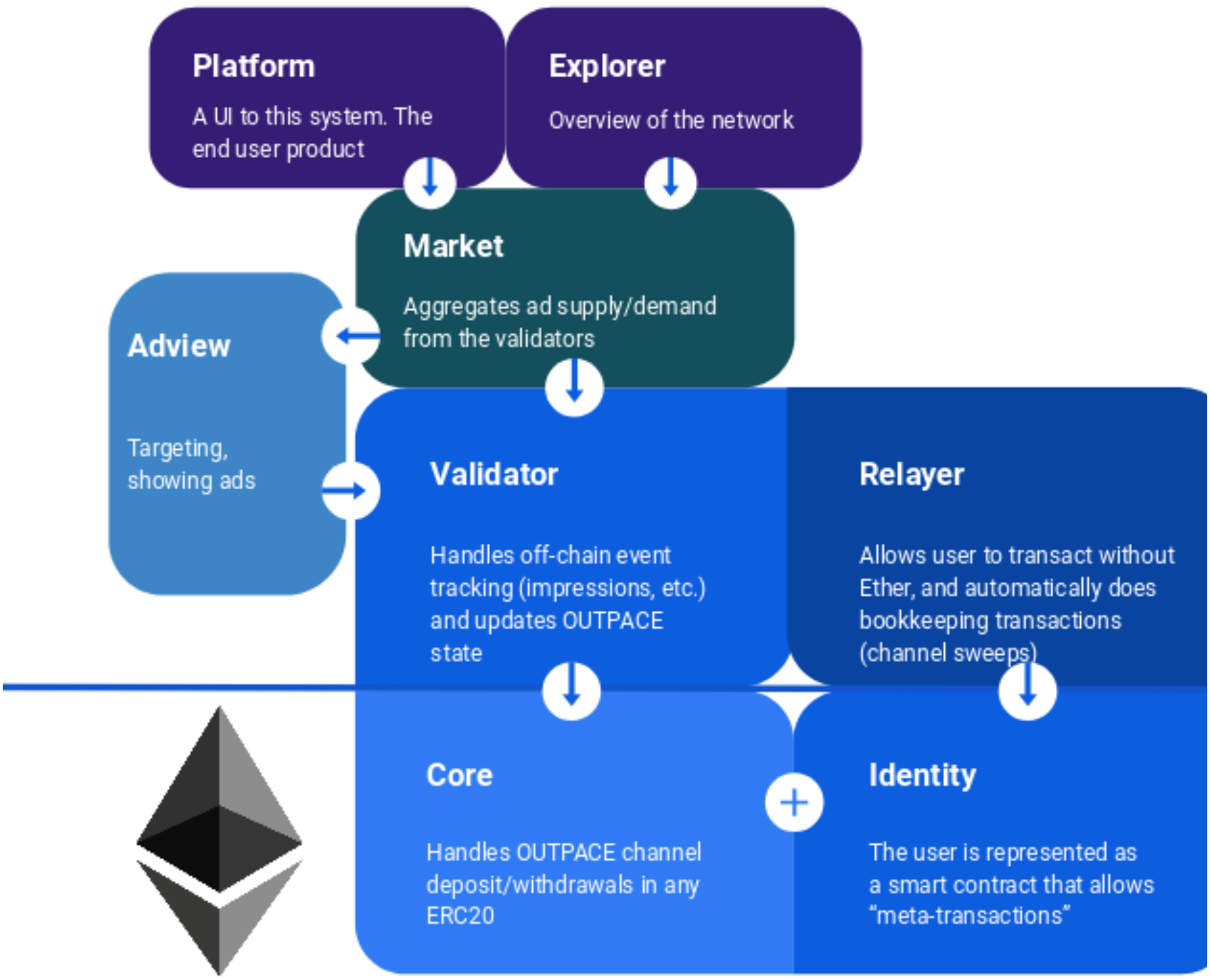
You can also stake through Binance and Huobi.

Automated buybacks

In May 2021, a mechanism was introduced in the Tom staking pool that uses the validator's fees to buy ADX and distribute it stakers.

Appendix

Basic visual representation



- The box-shaped platform and AdView are client-side software
- Round-shaped items represent parts of the AdEx peer-to-peer network (in practice, many validators and markets may
- The diamond shape represents another P2P network, in this case Ethereum

To keep the representation simple, we've omitted some components: for example, the Identity is used by publishers/advertis interact with the platform, and the Core runs on the Ethereum network itself. The Registry is a separate system, designed to platform users pick validators.

Preventing fraud/Sybil attacks

One of the main challenges of any digital advertising system is preventing fake or invalid impressions/clicks.

There are a few ways to mitigate that in AdEx:

1. Traditional adtech methods, such as IP whitelists/blacklists, as well as verifying publishers by their domain name

4. The AdView allows publishers to vouch for users of their website/app, for example if a user registers on your website a verifies a valid phone number; that allows users to gain reputation as "real" users, and therefore more conservative advertisers may define in their campaigns to only target users above a certain threshold
5. Publishers integrating the AdView may opt to show a captcha to users, the first time the user's cryptographic identity i created; this essentially means the user will solve the captcha once for all sites that integrate AdEx; they will need to so captcha again if they clear `localStorage` or change their browser.

It should be noted that such a system is, by definition, always gameable. AdEx tries to make it as hard as possible. We believ transparent reporting aspect of the system, combined with the "custom events", which allow you to track end results (e.g. registrations, purchases, etc.), ensure that the incentives for fraud are significantly reduced.

Scalability

Because impressions are tracked and rewarded off-chain, the only on-chain bottleneck of AdEx is depositing/withdrawing fu We think the current capacity of the Ethereum network is enough for thousands of advertisers and publishers, assuming they withdraw once every 2-3 weeks.

We do have a way to improve on-chain capacity as well: our OUTPACE payment channels are implemented with Substrate ar ready to be deployed on Polkadot. With possibility of interoperable blockchains designed only to handle OUTPACE channels scalability of AdEx is more or less unlimited.

Autonomous regulation

Ultimately, AdEx is completely censorship resistant since anyone can run their own Market and Platform and do whatever th want with them.

However, there are plenty of situations where you need control; for example, as a publisher, you may want your website to b of deceptive ads (malvertising).

The AdEx components provide multiple ways for the system to self regulate:

- Publishers can whitelist or blacklist advertisers or ad units;
- Advertisers can whitelist or blacklist publishers, topics (tags) or individual ad slots;
- Users can blacklist ad units, advertisers and even topics (tags).

Further down the line, reputation systems could be developed to make it easier for participants to push out low quality or deceptive ads.

Privacy of publishers/advertisers

There's nothing in AdEx requiring advertisers/publishers to identify themselves with anything other than a cryptographic ide Information that might reveal more (e.g. ad unit info, web addresses, creatives) is kept off-chain and and revealed between p only with explicit consent.

Furthermore, the full event history is distributed across validators/observers. Each validator will only collect the full event hist for the channels they're validating.

In other words, sensitive and valuable data is kept private to the parties that have accumulated it, and relationships between publishers/advertisers cannot be publically traced.

Anyone in the network can query any validators for events, but only for the events that they're involved in. For example, if yc publisher/advertiser/user, you can query all validators to get the events related to you.

Please note that the entire balance tree of each channel will be revealed to everyone at all times, (1) to allow earners (publish to observe it's validity and (2) it will be revealed on-chain anyway once everyone withdraws.

Privacy of the end-user

Privacy of end users is protected by not collecting any data at all, at any part of the system. Instead, we leverage contextual targeting.

Furthermore, we have moved the process of selecting an ad to show to the user's browser, which is essentially equivalent he bidding but with stronger privacy guarantees. This ensures that user data never needs to be exposed/revealed.

A further advantage to this approach is that the user can easily control what kinds of ads they see, without this being reveale third parties.

To ensure that infrastructure providers such as the Market and validators have no ability to collect data, AdView sends no identifiable user data when interacting with them. Furthermore, each ad campaign can be handled by different validators to that the validator operators cannot perform side-channel attacks. Finally, validators are required to implement the EFF's Do n track policy.

Rewarding end-users for attention

Through OUTPACE channels, it's possible that users are rewarded for certain events.

However, this is currently not something we intend to implement, mostly because it makes it might make it easier to perforn attacks and earn from fake traffic.

We do intend to implement this capability in the validator stack once we analyze the implications and risks. Once we've esta a model we're confident with, we will make this configurable through the `campaignSpec`.

In technical terms, everything needed to do this is there - every user signs an event with a keypair (which can be used for rec funds), OUTPACE channels allow easy micropayments, and users would be able to see their earnings and withdraw them thro the AdEx Lounge UI.

The fund flow would be: `advertiser -> {publisher AND user}`.

End-users paying for content

It is possible for users to pay for the publisher's content and therefore not see any ads.

This could be done in what we believe is a very fair way: by having users deposit funds (open an OUTPACE channel) through AdEx Lounge, and implicitly outbid advertisers for each ad they'd otherwise see.

That way, the ad space/attention is priced fairly by the market. This ensures that the users pay minimal amounts while still nc damaging the publishers' revenue.

The fund flow would be: `{user OR advertiser} -> publisher`.

A realistic way for this to work is for it to be implemented in an ad blocker, so that any ads that don't allow being implicitly c (not AdEx-enabled) would not appear at all.

To be explored further; including possible collaborations with ad blockers.

Real-time bidding / Header Bidding

Real-time bidding (RTB) is something we intentionally left out of the protocol, primarily because it relies on some details abc user being propagated around the network to the exchange.

From a scalability perspective, real-time bidding can be implemented using off-chain scaling solutions, however the privacy off is too big.

We don't consider this to be a major disadvantage as header bidding is very rapidly replacing RTB in the adtech industry any Header bidding is the process of pulling all the bids in the browser, evaluating them and then sending the preferred bids to exchange. In AdEx, there is no classic ad exchange, but what we do is even more convenient: we pull all information about d (campaigns, bids) in the browser, and directly select the bid depending on what we know about the user, therefore impleme targeting without revealing the user's profile.

In other words, **in AdEx, advertisers can bid for an impression in real-time**, but we do not implement traditional real-tim bidding.

See Flow and Bidding Process.

Oracle-based advertising

With the advancement of trust-minimized blockchain oracles, it is possible for AdEx to be used in a much wider set of use ca including, but not limited to:

- Ads in the physical world - e.g. highway banners, magazines;
- Video product placement;
- Influencer marketing, etc.

In those cases, OUTPACE will still be used, but the payments would be time-based ("time tick" event).

We believe that AdEx still offers benefits for those cases, mostly revolving around transparent auctions and payments.

To be explored further.

Harberger tax ownership model

There is a project that uses this model for ads right now, called Harberger ads.

In AdEx, it is possible to use the Harberger tax ownership model. However, due to the dynamic nature of digital advertising, practical for advertisers to fully buy and own ad spaces.

The way we envision the model working is by using the OUTPACE channels to pay rent, but paying rent on display time rathe on physical time ("time tick" event).

Role of AdEx Network OÜ

AdEx Network OÜ is a legal entity with the following primary responsibilities:

1) Fund and govern the development of the AdEx Protocol, with an emphasis of keeping it completely open-source, transpa and free of corporate agenda; 2) Profit from providing any additional services related to the AdEx Protocol, such as consulta related to integration of the protocol or running a SaaS for rentable AdEx validators. 3) Fund development of additional serv built around the AdEx Protocol; for this purpose, AdEx employs SmartCode, a software development company that also deve Stremio

Because of the open-source nature of the protocol, we do expect (and encourage) other entities interested in using it to join development/design over time.