Papyrus Docs Documentation

Release

Alexander Shvets

Jul 16, 2019

Contents:

Papyrus is the world's first fully comprehensive and highly scalable decentralized ecosystem for digital advertising which radically improves programmatic advertising stack to provide efficient, transparent and mutually beneficial environment for users, publishers, advertisers and decentralized application (dApp) developers using blockchain architecture.

CHAPTER 1

Overview

1.1 Market issues Papyrus solves

- Absence of transparency, incorrect incentives due to rebates from media companies.
- Long chains of middleman with large margin cuts between advertisers and publishers, only 30 to 40 cents of every digital media dollar are estimated to actually reach publishers and result in an ad showing up, according to ANA.
- Growth of ad-blocking software, 26% of US consumers use some sort of adblocking software in 2017, internationally, the loss of publisher revenue from ad blocking rose to $42 billion - up from $28 billion in 2016.
- Fraud traffic accounts to $6,5 billion losses in 2017 (~9% in desktop display, ~22% in video ads).
- Brand safety issues, for example, P&G cuts $140 million from digital ad spending recently due to brand safety and supply chain concerns.
- Viewability issues, 40%+ of ads served are out of view.

1.2 Blockchain solution for advertising market

- Fix transparency issues by creating decentralized storage of ad campaigns data with permissioned access and incentivizing market participants to store that data, decentralization protects data from manipulation
- Remove excessive and hidden budget cuts by allowing advertisers to make payments in tokens based on established payment conditions fixed in smart contracts, all supply chain participants receive payments according to smart contracts, smart contracts are executed using complete ad campaign information stored in decentralized storage
- Resolve brand safety, viewability and fraud issues by connecting auditors / antifraud vendors / attribution providers that verify ad traffic according to smart contracts and put verification information into decentralized storage, ad campaign payments are made automatically according to verification results, dispute resolution happens automatically

- Transition from ad blockers to value exchange with end users by providing tools to publishers to construct dialogue with users on value exchange, users can turn off ads and pay content subscription fees or engage with ads and receive compensations in the form of content access or tokens
- Less paperwork and organizational expenses on payments reconciliation and disputes resolution due to usage of token for payments and automation of processes via smart contracts

CHAPTER 2

Papyrus scanner

2.1 General

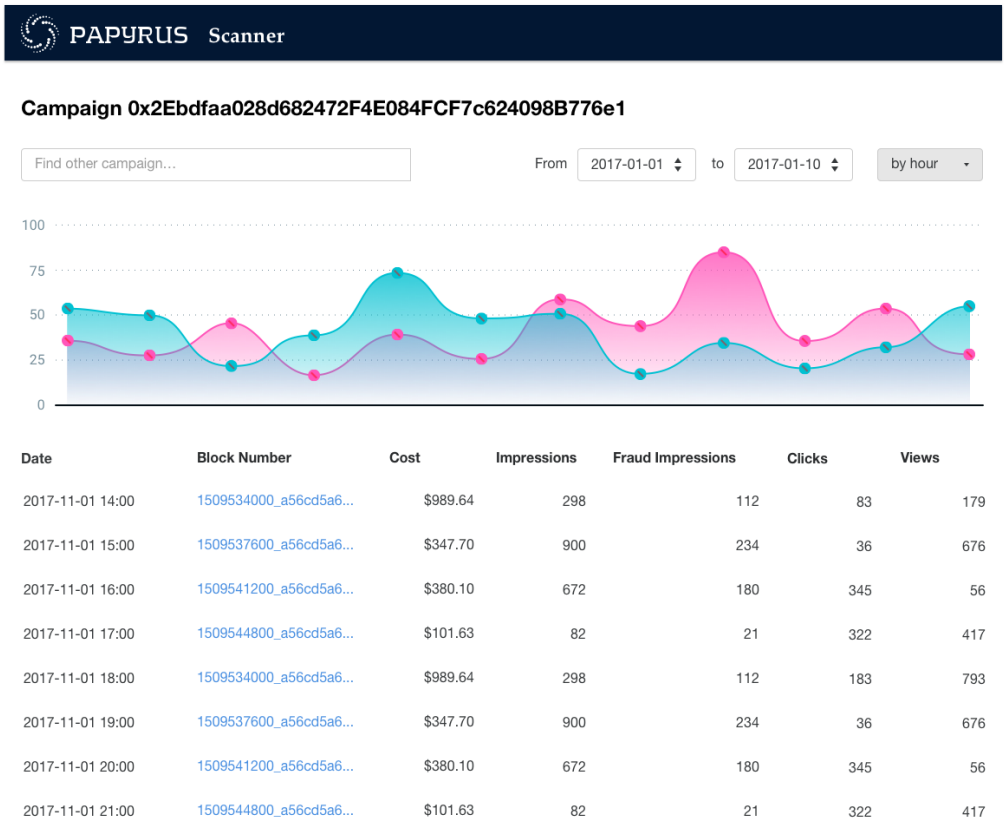
Papyrus scanner is a software solution to view Papyrus blockchain. It enables to any participant (advertisers, publishers, vendors) to see ad campaigns they pariticpate into.

Papyrus scanner provides information about ad campaign smart contracts and aggregates validated statistics information. The scanner shows aggregated statistics blocks where you can find

- actual cost paid to SSP and other vendors;
- amount of fraud impressions;
- actual clicks and views count.

For example, you can see that there were 1,975 impressions, but in fact only 1,545 were real not fraud ones. And you can view real clicks count and impressions. More important that you paid only for these real impressions.

The agency fee and other vendor fees are included in reporting according to the campaign contract, so you can be sure that there are no opaque markups.



2.2 Scanner API

2.2.1 Read campaign list

GET /campaigns
Returns a list of campaigns.

Request example

```
GET /api/v1/campaigns HTTP/1.1
Host: scanner.papyrus.global
Accept: application/json
```

Response example:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "id": "2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824",
    "name": "Campaign name"
    "advertiser_id":
↪"0a526a90a85596dcb3669fd86963422969edbbf7c4752492d780b78e6355d4ee",
    "start_date": "2018-01-01",
```

```
    "end_date": "2018-01-31",
    "budget": "10000000000",
    "maximum_cpm": "10000000",
    "ssps": [
      {
        "id": "007d831ea2e8e1d080b31e33c50b89ea07f9b694bccad998c8cf5cb1a087f889",
        "fee_percent": "0.1"
      }
    ]
    "auditors": [
      {
        "id": "c5a62ce3fa7f6d86af0009389ccd815277691ea64da0c5c98e302bb13dd59248",
        "fee_percent": "0.05"
      }
    ]
  }
]
```

Query Parameters

- **key** (*String*) – participant_key, required
- **campaign_id** (*String*) – campaign filter
- **advertiser_id** (*String*) – advertiser filter
- **dsp_id** (*String*) – dsp filter
- **ssp_id** (*String*) – ssp filter
- **auditor_id** (*String*) – auditor filter

2.2.2 Read campaign statistics

GET /statistics

Returns campaign statistics.

Request example

```
GET /api/v1/statistics?campaign_
↪id=2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824 HTTP/1.1
Host: scanner.papyrus.global
Accept: application/json
```

Response example:

```
HTTP/1.1 200 OK
Content-Type: text/javascript

[
  {
    "date": "2017-12-12",
    "block_number": "1511718000_
↪496aca80e4d8f29fb8e8cd816c3afb48d3f103970b3a2ee1600c08ca67326dee"
    "cost": "12340000",
    "impressions": "1234",
    "fraud_impressions": "321",
    "clicks": "56",
```

```
    "views": "77",
    "ssps": [
       {
         "id": "007d831ea2e8e1d080b31e33c50b89ea07f9b694bccad998c8cf5cb1a087f889",
         "fee": "1234000"
       }
    ]
    "auditors": [
       {
         "id": "c5a62ce3fa7f6d86af0009389ccd815277691ea64da0c5c98e302bb13dd59248",
         "fee": "617000"
       }
    ]
  },
  {
    "date": "2017-12-12",
    "block_number": "1511723000_
↪6d0b07ee773591f2a1b492d3ca65afdefc90e1cadfcc542a74048bb0ae7daa27"
    "cost": "43210000",
    "impressions": "4321",
    "fraud_impressions": "789",
    "clicks": "123",
    "views": "135",
    "ssps": [
       {
         "id": "007d831ea2e8e1d080b31e33c50b89ea07f9b694bccad998c8cf5cb1a087f889",
         "fee": "4321000"
       }
    ]
    "auditors": [
       {
         "id": "c5a62ce3fa7f6d86af0009389ccd815277691ea64da0c5c98e302bb13dd59248",
         "fee": "2160500"
       }
    ]
  }
]
```


Query Parameters

- **key** – participant_key, required
- **campaign_id** – campaign filter, required
- **date_from** – date filter
- **date_to** – date filter

CHAPTER 3

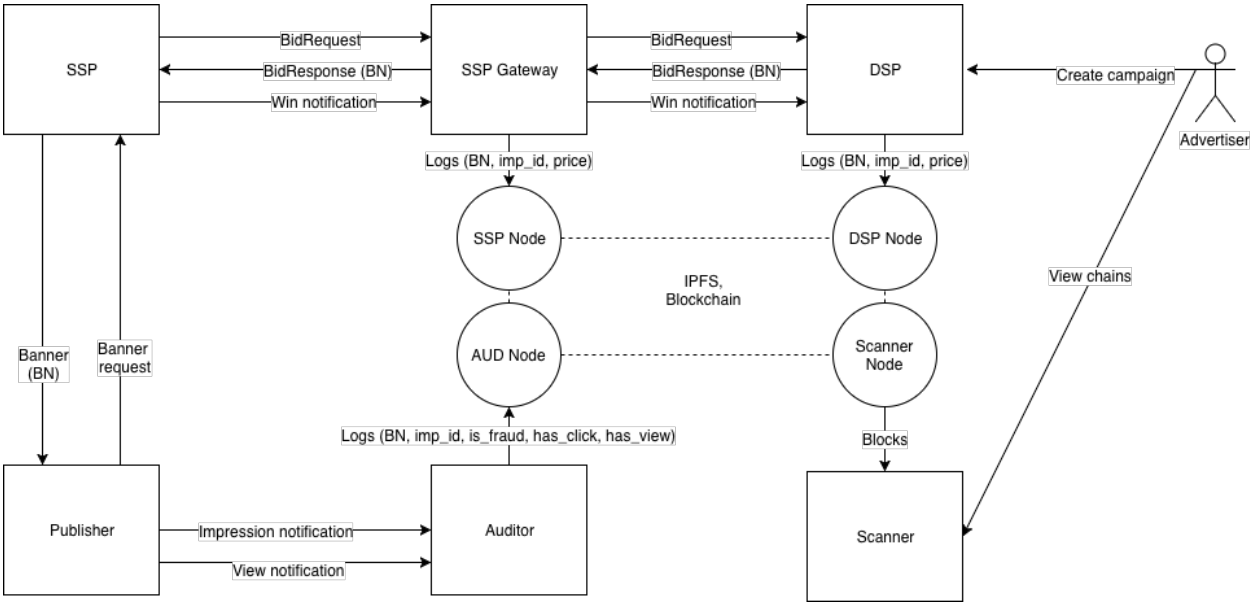
SSP integration

The most easiest participant for an integration into the Papyrus ecosystem is SSP.

There are 3 ways of SSP integrations.

1. Papyrus deploys SSP gateway with its own blockchain node and SSP connects to this gateway through API.
2. Papyrus installs the node and SSP sends logs to this node.
3. SSP installs its own node and communicates with it internally.

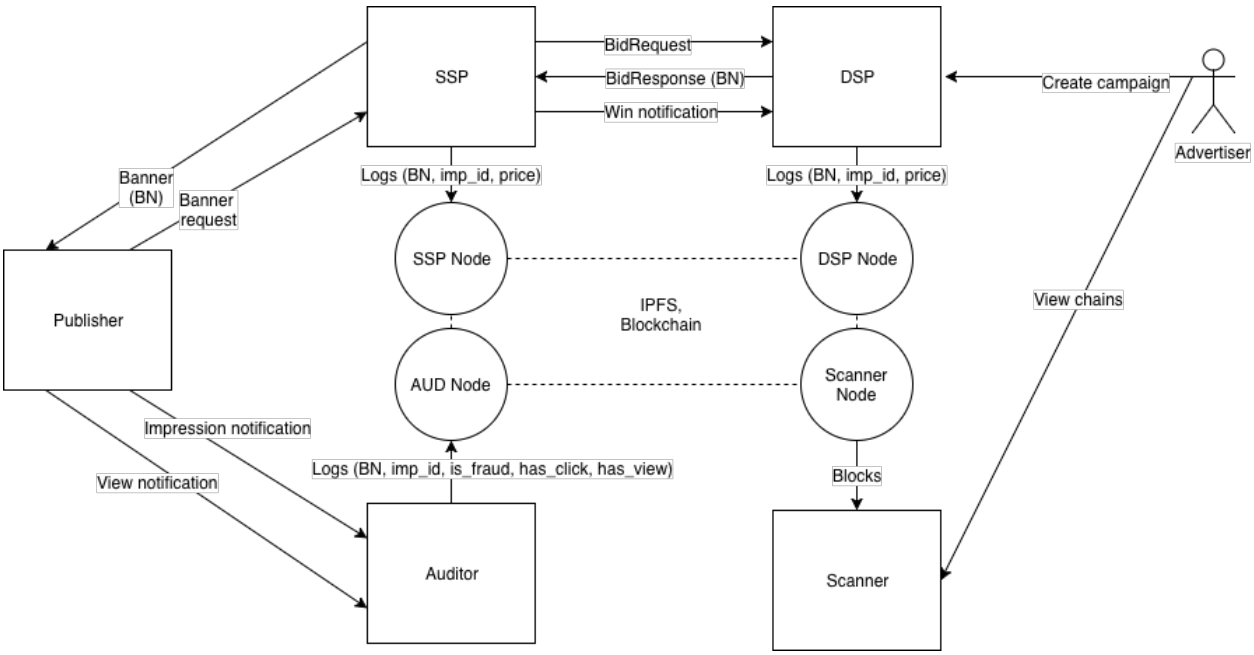
3.1 Papyrus SSP gateway



For basic integration Papyrus can deploy SSP gateway by its own team. In this case SSP can connect SSP gateway like any other DSP.

SSP gateway is the system which passes Bid Requests and Win Notifications from integrated SSP to connected DSPs. SSP just sends all requests to SSP gateway instead of DSP and gateway does all necessary job to put log records into decentralized storage. The auction is holded by SSP itself and gateway just records the winner record. SSP Gateway has to send log record to Channel Node after receiving Win Notification. The record contains block number, impression_id and the winning price.

3.2 Papyrus node



This case is similar to previous, but doesn't require SSP gateway. Papyrus team just deploys Papyrus blockchain node and SSP sends log records into this node through gRPC.

The main difficulty that SSP has get block number from Bid Response. This block number is written in *ext.blocknumber* field of Bid object.

SSP has to send message on Win Notification generation. The format of gRPC message is presented below.

```
// Main channel interface
service StateChannel {
    // Creates or updates outgoing channel with given participant
    rpc RegisterTransaction(RegisterTransactionRequest) returns␣
↪(RegisterTransactionResponse);
}

// Registers transaction
message RegisterTransactionRequest {
    // sender address in HEX, from config
    string sender = 1;
    // block_number, from Bid Response
    int64 block = 3;
    // encoded message, format below
    bytes data = 4;
```

```
    // EC signature by sender's key, from congif
    bytes signature = 5;
}


message PapyrusWinNotification {
    string imp_id = 1;
    // price in token * 10^18
    int64 price = 2;
}
```

3.3 SSP node

This case is similar to previous, but in this case SSP has to install its own blockchain node. Papyrus team distibutes SSP node as docker image with instruction provided. The link to the distro will be published later.

CHAPTER 4

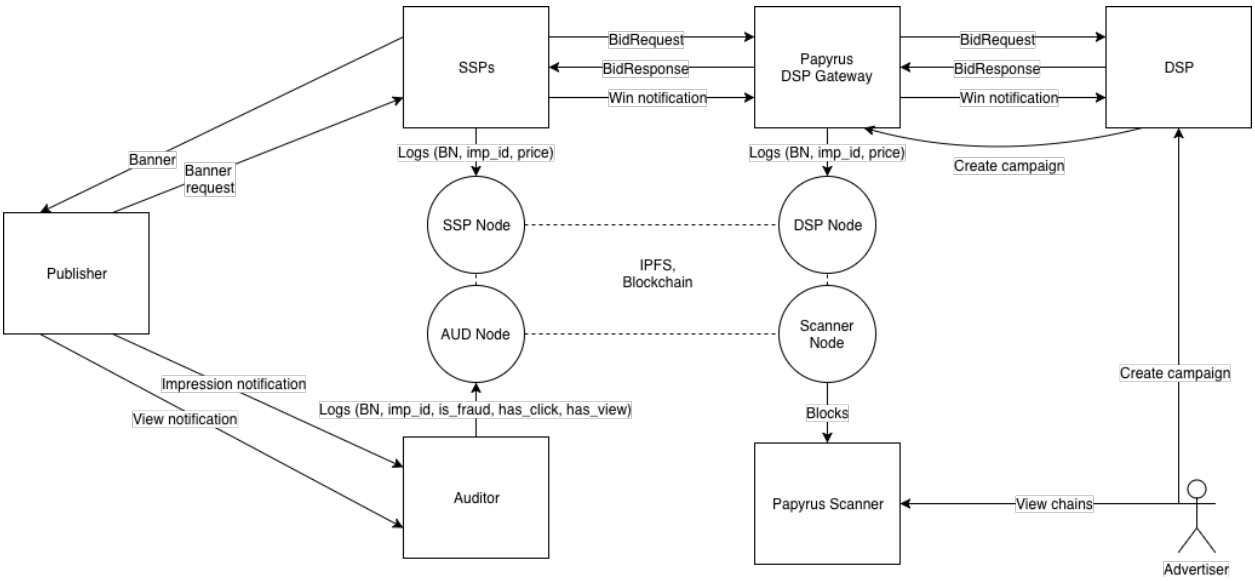
DSP integration

DSP is the key participant in the ecosystem because DSP is in charge of creating ad campaign smart contracts

There are 3 ways of DSP integrations.

1. Papyrus deploys DSP gateway with its own blockchain node and DSP connects to this gateway through API.
2. Papyrus installs the node and DSP sends logs to this node. Also DSP has to create smart contracts through this node.
3. DSP installs its own node and communicates with it internally.

4.1 Papyrus DSP gateway



To make integration easier Papyrus team can deploy DSP gateway to include DSP into Papyrus ecosystem. This gateway acts like any additional SSP, but actually it does all necessary job necessary for blockchain working and resends messages to connected SSPs and in other way.

Integration of DSP gateway differs from SSP integration in 2 points:

1. DSP has to create smart contract in Papyrus blockchain for every ad campaign.
2. DSP has to generate block number for every impression it put bid on.

DSP gateway provides an API method to creae smart contracts.

POST /campaigns

Creates a smart contract for campaign.

Request example

```
POST /api/v1/campaigns HTTP/1.1
Host: scanner.papyrus.global
Accept: application/json
Content-type: application/json

{
        "name": "Campaign name"
    "advertiser_id":
↪"0a526a90a85596dcb3669fd86963422969edbbf7c4752492d780b78e6355d4ee",
    "start_date": "2018-01-01",
    "end_date": "2018-01-31",
    "budget": "10000000000",
    "maximum_cpm": "10000000",
    "ssps": [
      {
        "id": "007d831ea2e8e1d080b31e33c50b89ea07f9b694bccad998c8cf5cb1a087f889",
        "fee_percent": "0.1"
      }
    ]
    "auditors": [
      {
        "id": "c5a62ce3fa7f6d86af0009389ccd815277691ea64da0c5c98e302bb13dd59248",
        "fee_percent": "0.05"
      }
    ]
}
```

Response example:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "id": "2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824"
  }
]
```

Advertisers, SSP and Aufitor IDs will be provided by decentralized registry in the future, now this registry is provided by Papyrus team and should be saved locally.

The result ID is the smart contract ID and should be saved for the further usage.

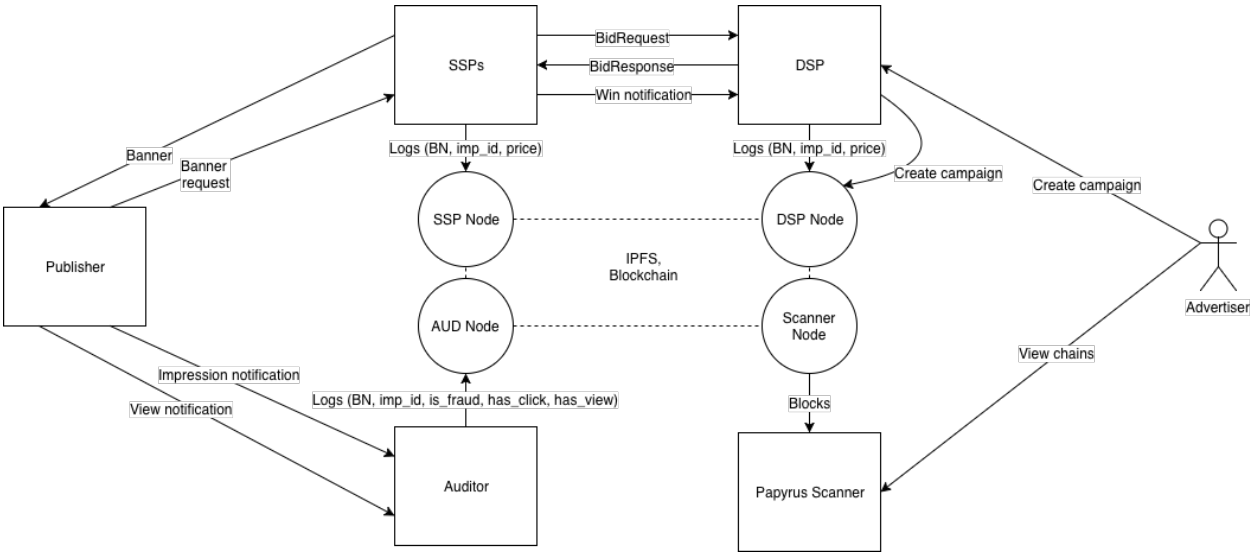
The contract ID is used in block generation.

block_number = current_timestamp + '_' + contract_id

A new block number should be generated for the contract every hour. This time window can be smaller in case of big number of messages in block.

Generated block number has to be included in *ext.blocknumber* field of Bid object.

4.2 Papyrus node



This case is similar to previous, but doesn't require DSP gateway. Papyrus team just deploys Papyrus blockchain node and DSP sends log records into this node through gRPC.

DSP has to send log message right after receiveing Win Notification. The format of gRPC message is presented below.

```
// Main channel interface
service StateChannel {
    // Creates or updates outgoing channel with given participant
    rpc RegisterTransaction(RegisterTransactionRequest) returns␣
↪(RegisterTransactionResponse);
}

// Registers transaction
message RegisterTransactionRequest {
    // sender address in HEX
    string sender = 1;
    // block_number
    int64 block = 3;
    // encoded message
    bytes data = 4;
    // EC signature by sender's key
    bytes signature = 5;
}


message PapyrusWinNotification {
    string imp_id = 1;
    // price in token * 10^18
```

```
    int64 price = 2;
}
```

Also, DSP has to create smart contract using the node API. This API will be specified later.

4.3 DSP node

This case is similar to previous, but in this case DSP installs its own blockchain node. Papyrus team distibutes blockchain nodes as docker images with instruction provided. The link to the distro will be published later.

CHAPTER 5

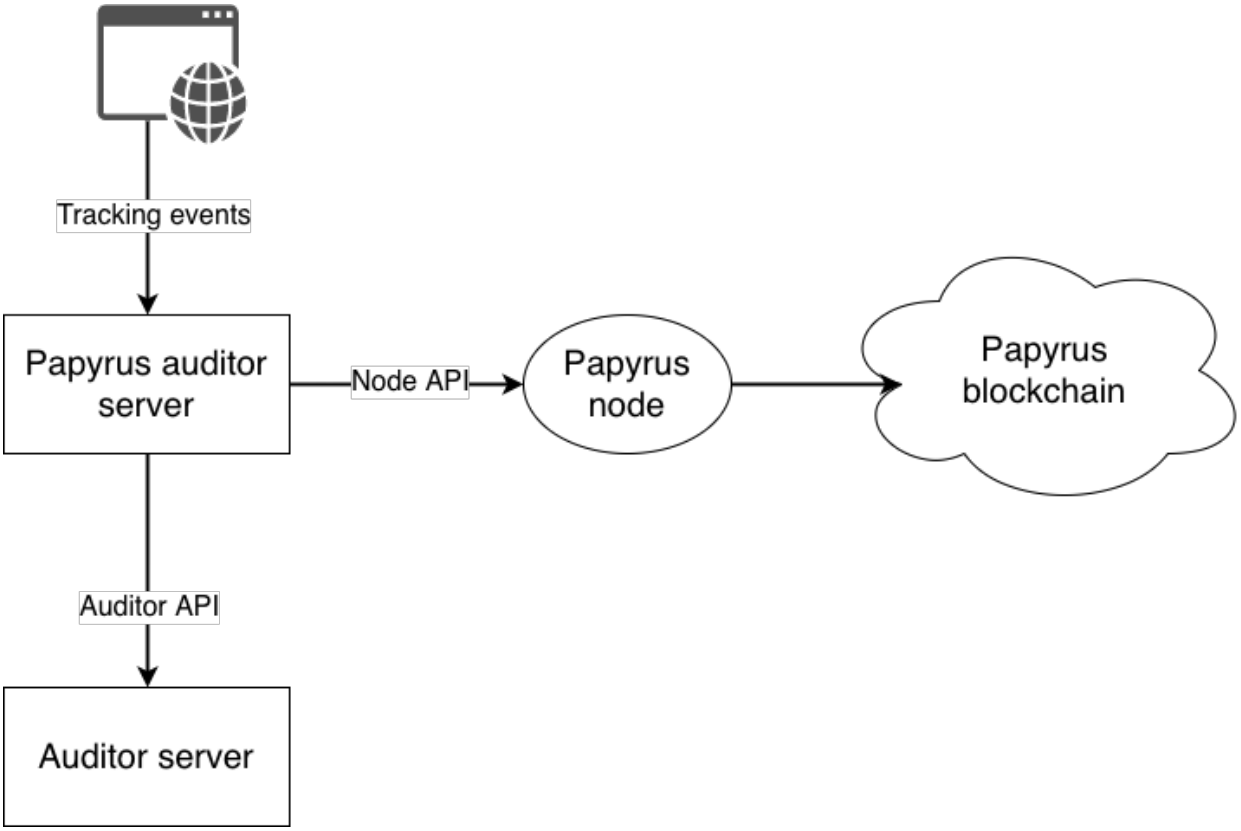
Auditor integration

Auditor integration is the major feature of the Papyrus ecosystem. It enables to make payments only on real impressions and other events.

There are several ways of auditor integrations.

1. Papyrus deploys log server with its own blockchain node and connects to auditor through API.
2. Papyrus installs the node and auditor sends logs to this node.
3. Auditor installs its own node.

5.1 Papyrus log server and Papyrus node



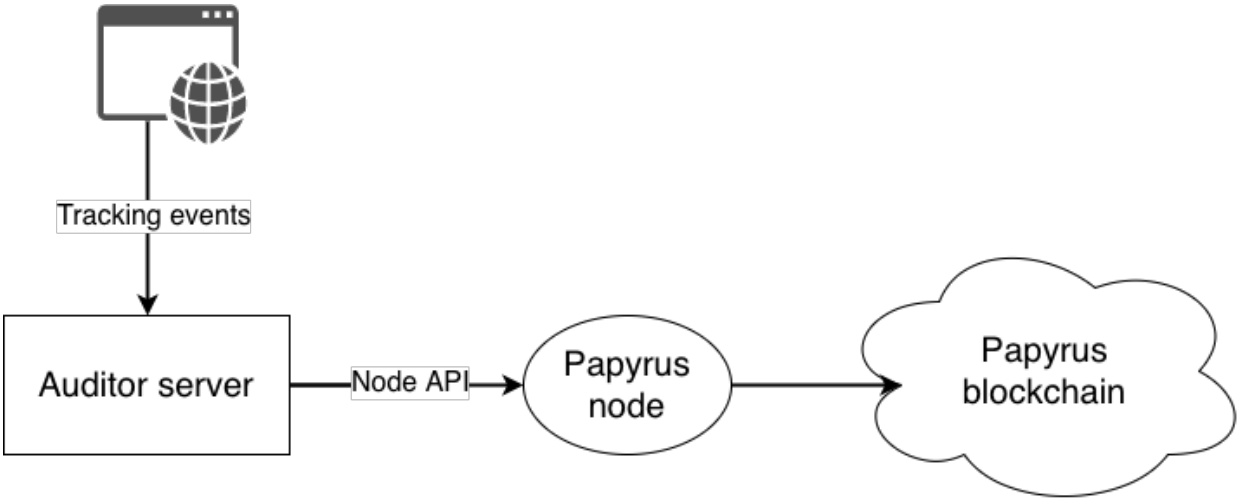
This is the easiest way to connect auditor to Papyrus ecosyste, because Papyrus development team setup and maintain this integration on their side.

But in this case auditor has to provide an API to check requests.

API can vary for different vendors, but it must support at least the following parameters:

- request type
- user IP
- user agent
- page URL
- campaign ID

5.2 Auditor log server and Papyrus node



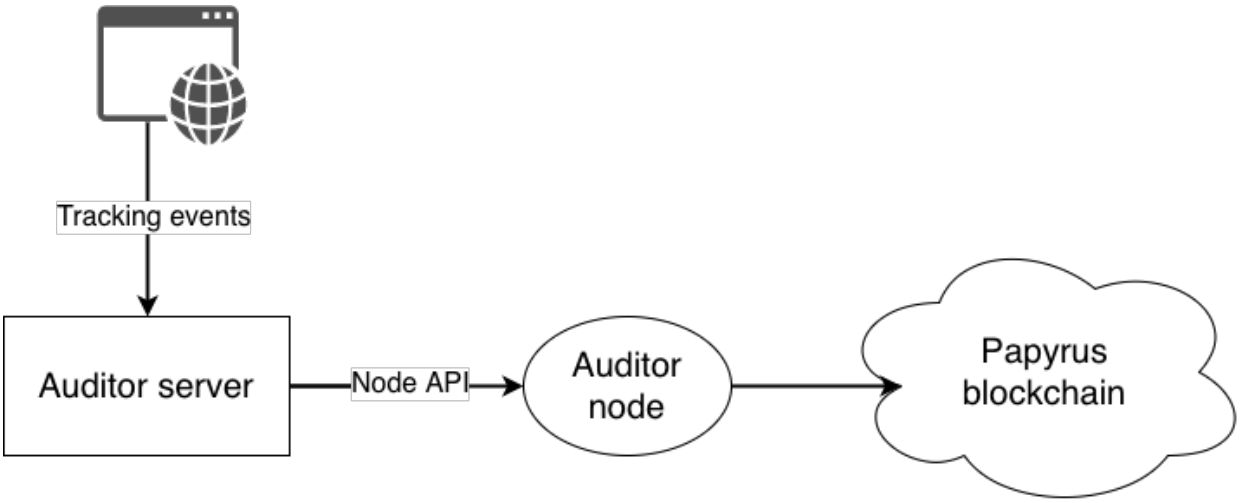
This case has medium difficulty for auditor. It requires that auditor process events itself and works with blockchain node, but blockchain node is maintained by Papyrus team.

Papyrus blockchain node has gRPC API with the following message structure

```
// Main channel interface
service StateChannel {
    // Creates or updates outgoing channel with given participant
    rpc RegisterTransaction(RegisterTransactionRequest) returns␣
↪(RegisterTransactionResponse);
}

// Registers transaction
message RegisterTransactionRequest {
    // sender address in HEX
    string sender = 1;
    // channel contract address in HEX
    string channel = 2;
    // block_number
    int64 block = 3;
    // encoded message
    bytes data = 4;
    // EC signature by sender's key
    bytes signature = 5;
}

message RegisterTransactionResponse {
}

message PapyrusAuditorFeedback {
    string imp_id = 1;
    bool flags = 3;
    enum Flags {
        EMPTY=0;
        FRAUD=1;
        VIEW=2;
        CLICK=4;
    }
```

```
}
```

5.3 Auditor log server and node



This case is the most diffucult for auditor, but it is the most effective solution. Auditor has to setup its own blockchain node and send logs to its API as in the previous case.

The Papyrus node is distibuted as docker container. We will describe installation guide soon.

CHAPTER 6

Advertiser and publisher nodes

To ensure that the process is valid other participants like advertisers and publishers can install their own nodes.

This section will describe installation process for these participants.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

HTTP Routing Table

/campaigns

/statistics