Wolk SWARMDB: Decentralized NoSQL and Content+Data Marketplace

October 1, 2017

Wolk Inc. and The Wolk Foundation

Abstract

Ethereum SWARM is an open source project of the Ethereum Foundation being adapted and extended by Wolk to support: (1) SWARMDB, a decentralized NoSQL implementation to store and retrieve id-column-value combinations in the style of Google BigTable / Apache HBase; (2) decentralized content + data marketplaces, where providers are paid for content + data delivered through SWARM. This white paper outlines Wolk's goals on both efforts using Ethereum-backed ERC20 WOLK tokens, offered in a WOLK Token Generation Event ("Wolk Token Generation Event" or "TGE") for contributors to develop the Wolk Ecosystem where storage and bandwidth providers earn WOLK for this decentralized NoSQL and content/data marketplace. This is expected to have wide applicability for: (a) decentralized applications, who will pay WOLK to store and retrieve record data (b) content/data providers, who will earn WOLK for providing highly desirable content/data stored in truly decentralized data exchange -- without anyone acting as an intermediary. We outline how a decentralized index and SWARMDB interfaces support these efforts and the development of the Wolk ecosystem.

1. Wolk Mission and Core Value Proposition

Wolk's mission is to create open data systems for the world. Wolk aims to do this by adapting Ethereum SWARM, an open source project of the Ethereum Foundation. In this white paper, we describe how Wolk is adapting and extending SWARM to support the development of:

(1) SWARMDB, a decentralized NoSQL implementation to store and retrieve id-column-value combinations in the style of Google BigTable / Apache HBase;

(2) decentralized paid content / data marketplaces, where content of files and data in SWARMDB tables can be obtained using WOLK.

A new ETH-backed WOLK token is used to power incentives to SWARMDB for both core use cases, and this paper describes parameters for a WOLK Token Generation Event.[1]

Background

The Ethereum SWARM project has a functioning implementation of a Kademlia-based peer-to-peer storage protocol backed by ERC20 token-based incentives for *bandwidth* and *storage*. Files are chunked into a Merkle tree where the top level root hash identifies the file in a collision-resistant way.

Chunks are forwarded peer-to-peer to nodes where the closest nodes store the chunks using Kademlia's well-understood XOR distance metric and support $\log_2(n)$ storage and retrieval. When a Merkle tree root hash requests a file, in parallel, the SWARM requests a set of chunks, and each chunk is retrieved from SWARM nodes forwarding on chunk requests to the closest nodes that contain the chunk, passing chunks back to the original requesting node.

SWARM's accounting system is done with ERC20 tokens, and in Wolk's (currently private) SWARM, the ERC20 token is WOLK (rather than ETH, as in SWARM's current POC docs/implementation). The basics of SWARM's incentivization model are as follows:

- For Bandwidth, one SWARM node compensates one of its direct peers when its balance exceeds a specific threshold, and stored on the Ethereum blockchain. Each node maintains an internal count of "Checks" that can be cashed in from each of its peers. This incentivizes delivery of highly popular chunks of data where:
    - If a content/data buyer/node gets a lot of data, they pay WOLK for it
    - If a content/data supplier/node delivers a lot of data, they earn WOLK
- For Storage, SWARM incentivizes *long-term* storage (specifically of rarely accessed chunks) with nodes that register with the SWEAR contract that implements a WOLK-based security deposit secured on the Ethereum blockchain. Registered nodes sell receipts for chunks received, where receipts are used in the challenge-response "court" system of SWINDLE contracts. A system of guardians implement the court system, enabling data uploaders to upload and disappear and effectively check whether chunks are still available by the nodes swear. When nodes cannot provide (or can) proof of custody of chunks, the guardians/judges award the deposit to the data requester (or not).

[1] Please refer to the WOLK Token Term Sheet for information on who may participate in the Token Generation Event.

While SWARM was not conceived to support record storage or incentives for providing data+content, the mature infrastructure of chunks, manifests, and token-based incentives offers a solid base for SWARM to be extended along these lines.

Earlier versions of Wolk's data exchange focused on developing only a *partially* decentralized architecture; where Wolk was an intermediary between buyer and seller and Wolk Inc would earn up to 20% of WOLK as data buyers used APIs to access data. This was not a *fully* decentralized design, because Wolk continued to act as an intermediary. Initially, we were aiming for BigTable as a cache for SWARM data, but as our design for SWARMDB developed in Summer 2017, it became clear that SWARMDB could support a completely decentralized design, and eliminate Wolk's position as an intermediary.

2. SWARMDB

Wolk's design for SWARMDB is one where decentralized file storage is repurposed to support the organization of "schemaless tables" (called "NoSQL"), where table records are indexed solely by a single primary key. Each table record can have any number of unspecified column qualifiers and columns and values associated with those columns, obeying a hierarchy of:

```
table owner ("0x7289…")
        table ("account")
                id ("jsmith1234@gmail.com")
                        column qualifier ("profile")
                                column (profile:face)
                                        value ("\0x48\0xa7\0xff...")
```

Table owners are specified by their Ethereum address (e.g. "0x7289…") interacting with a SWARMDB interface, which must hold WOLK tokens to store and retrieve rows from a SWARMDB table. Rows in the table are indexed by id (e.g. "jsmith1234@gmail.com"). A column qualifier (e.g. "profile") allows for restricting the number of column-value combinations for efficient retrieval.

A specific *{ owner, table, id }* combination is mapped to a SWARM manifest holding the columns and values (more precisely, SWARM hashes of the values) for that id. Record-level read permissions are one of these 3 possibilities:

1. Private – the data can only be read by the owner of the data
2. Permissioned – the data can be read with permissions, using proxy re-encryption methods
3. Public – the data can be read by everyone

The full design for the permissioned case using decentralized Key Management Systems and proxy re-encryption we leave to a subsequent technical note; planned design with Wolk partner Nucypher is underway but needs coordination with SWARM's efforts to have chunk-level encryption. Concerning write permissions, the current working design is that records of the table can only be written to by the table owner. Designs to support public writes by multiple data suppliers for the same id is currently underway.

2.1 SWARMDB Interfaces

Wolk's implementation of SWARMDB exposes:

1. a command line interface
2. HTTP API
3. Go interface modelled after Google BigTable's interface.

so that content/data requesters and suppliers can read and write records into SWARM. The SWARMDB client has a authenticated Ethereum account that has WOLK credited or debited to it for all gateway operations.

In these interfaces, each time content/data suppliers upload content/data onto SWARM, they may specify their bounty amount, in WOLK. This bounty is the amount a requester must "pay" to access the file or record. For now, we assume optimistic retrieval of the entirety of content/data for now (e.g. access to less than 100% of the chunks of a file / record results in 100% of the bounty), and that failure to retrieve data are prosecuted by SWARM's court system.

When a node requester wishes to pay a bounty, the maximum WOLK bounty the node requester is willing to pay is sent in the form of a "transaction receipt" representing a promise to pay up to that amount WOLK if the top level data is returned. This transaction receipt is passed peer-to-peer until the data chunk is found at some node leaf. If this node leaf has a WOLK bounty attached to it specified in an earlier write operation (either for a specific file or a record's id-column) then that amount may be collected by the end node from the buyer based on the transaction receipt. This bounty for delivery is on top of the storage and bandwidth incentives on SWARM. We expect to evolve this paid content / data delivery mechanism significantly in the near future.

2.1.1 Command Line Interface

A command line interface "swarmdb" enables writing table/column/value combinations for:

- Writing Public records with column + value specified by local file, where the presence of a bounty makes the record public:

```
        swarmdb up --table account --id jsmith1234@gmail.com -column profile:photo
-file john-selfie.jpg -bounty 2.0
       [ { "column": "profile:photo", "tx":
"0x0bbc266eae70eaf962eab2ddb3be48a0975771b44a1d4f7cf2e227fbf76916b4" } ]
```

- Writing Private records with column + value specified on commmand line where the lack of a bounty makes the record private.

```
       Example: swarmdb up --table account --id jsmith1234@gmail.com -column
stats:uploadCount -value 723
       [ { "column": "stats:uploadCount", "tx":
"0x3be48a00bbc266eae7076916b4e975371baf962eab2ddb44a1d4f7cf2e227fbf" } ]
```

- Writing Public content specified by local file, where the bounty makes the data public and the lack a table specification adds to a generic "content" table shared by all

```
       Example: swarmdb up --id "Beethoven-Ode to Joy" --file ode-to-joy.mp4 -bounty
0.001
```

- Reading content of table keyed by id:

```
       Example: swarmdb --table account --id jsmith1234@gmail.com
       [{"column":"profile:name", "value": "john smith"},
       {"column":"stats:lastUpdatedTS", "value": "1424712341"},
       {"column":"stats:uploadCount", "value": "723"},
       {"column":"profile:photo", "value": "0x1423..."} ]
```

2.1.2 HTTP API

The previous command line interface maps into the HTTP API calls below:

GET http://localhost:8500/swarmdb:/tbl/id/column
Retrieve from the table `tbl` a specific column `column` value for the id `id`, retrieved in JSON form

Request:
http://localhost:8500/swarmdb:/account/jsmith1234@gmail.com/profile:photo

Response:
```
        [{"column":"profile:photo", "value": "0x1423"}]
```

GET http://localhost:8500/swarmdb:/tbl/id
Retrieve from the table `tbl` *all* the columns values for the id `id`, retrieved in JSON form

Request:
http://localhost:8500/swarmdb:/account/jsmith1234@gmail.com

Response:
```
        [{"column":"profile:name", "value": "john smith"},
        {"column":"stats:lastUpdatedTS", "value": "1424712341"},
        {"column":"stats:uploadCount", "value": "723"},
        {"column":"profile:photo", "value": "0x1423..."} ]
```

GET http://localhost:8500/swarmdb:/tbl/id?cq=cqfilter
Retrieve all the columns values for the id where each column is part of the column qualifier family in "cq", retrieved in JSON form.

Request:
http://localhost:8500/swarmdb:/account/jsmith1234@gmail.com?cq=stats

Response:
```
        [ {"column":"stats:lastUpdatedTS", "value": "1424712341"},
         {"column":"stats:uploadCount", "value": "723"} ]
```

POST http://localhost:8500/swarmdb:/tbl/id/column
The POST request writes a (table, id, column, value). The value for the column is specified in the body of the POST. SWARMDB returns the hash of the record along with the transaction receipts

Request:
http://localhost:8500/swarmdb:/account/jsmith1234@gmail.com
```
        [{"column":"profile:name", "value": "john smith", "bounty": .001},
        {"column":"profile:photo", "value": "0x1423...", "bounty": .01} ]
```
Response:
```
        [ {"column": "profile:name", "tx":
"0x6492d8487c03c2f33a8dc83c2980e818485c3033486920e26af5554e909bc459" },
        { "column": "profile:photo", "tx":
"0x0bbc266eae70eaf962eab2ddb3be48a0975771b44a1d4f7cf2e227fbf76916b4" } ]
```

POST http://localhost:8500/swarmdb:/tbl/id
The POST request expects a full JSON structure of all column and values.

2.1.3 Go Interface

The SWARMDB Go interface is modelled on Google BigTable's interface.

1. To read a *single* row from SWARMDB, use the ReadRow method of SwarmDB.Table:

```
swarm_table := swarm_client.Open("account")
// "jsmith1234@gmail.com" is the entire row key
row, err := swarm_table.ReadRow(ctx, "jsmith1234@gmail.com")
...
```

2. To read *multiple* rows from SWARMDB, use the ReadRows method of SwarmDB.Table to get multiple rows. A RowRange specifies a virtual portion of a table. A RowFilter such as FamilyFilter limits the data that is returned to specific column families.

```
swarm_table := swarm_client.Open("account")
// Read rows starting with "jsm", but only columns with the "profile"
column qualifier
prefix_range := swarmdb.PrefixRange("jsm")
err := swarm_table.ReadRows(swarmdb_context, prefix_range,
func(swarm_row Row) bool {
   // do something with swarm_row
   return true // keep going
}, swarmdb.RowFilter(swarmdb.FamilyFilter("profile")))
...
```

3. To write a row to SWARMDB, use Mutation to build up one or more column-value operations on a table, and then use the Apply methods on the Table to execute the operations. For instance, to set a couple of cells in a table,

```
swarm_table := client.Open("account")
mut := swarm_table.NewMutation()
mut.Set("profile", "name", bigtable.Now(), []byte("John Smith"),
.001)
mut.Set("stats", "lastUpdatedTS", bigtable.Now(),
[]byte("1424712341"), 100)
err := tbl.Apply(ctx, "jsmith1234@gmail.com", mut)
…
```

The WOLK Bounty is set in the Mutation record.

2.2 SWARMDB's Index

One central task of a database is for fast record lookup by a table's primary key. In SWARMDB, the challenge is to be able to quickly look up any id data. For our POC, every value ({ owner, table, id }) is hashed using a simple ":" separator using the 256-bit SHA3 hash of "owner:table:id" e.g.
`0x728781E75735dc0962Df3a51d7Ef47E798A7107E:account:jsmith1234@gmail.com` =>
`2477cc8584cc61091b5cc084cdcdb45bf3c6210c263b0143f030cf7d750e894d`
`...`
Currently, SWARM's POC stores the mapping between ids using an Ethereum Patricia Tree where the mapping between the hashed ids

```
Record 1: 2477cc8584cc61091b5cc084cdcdb45bf3c6210c263b0143f030cf7d750e894d
=>
SWARM HASH:
b5cc084cdcdb45bf3c622477cc8584cc6109110c263b0143f030cf7d750e894d
...
```

To store a record (owner, table, id) in SWARM with data D:

1. Content Storage: Build Merkle tree for data D, store in SWARM, getting back swarm hash `H(D)` (based on the content D) and transaction receipt `TXR(D)`.
2. Index Storage: Compute hash `HID(owner, table, id)`, and update Patricia Tree using Kademlia routing.

To retrieve a record (owner, table, id):

1. Index Retrieval: Compute hash `HID(owner, table, id)`, and obtain SWARM hash of record from Patricia Tree via Kademlia routing
2. Content Retrieval: Build Merkle tree from SWARM hash, reconstructing missing data

When a record is updated (any column, any value for any column), the SWARM manifest changes. If a large number of updates are committed to the same record, at the gateway it is a simple optimization to batch process and only update the (owner, table, id). Another potential optimization is to keep "small" content values (much less than a chunk, ie less than 32 bytes) directly in the index in place of the SWARM hash; this can be done with the Ethereum RLP scheme.

Clearly, SWARMDB makes no attempt for consistency and so decentralized applications must only use SWARMDB for records that do not change much more often than every few seconds.

A key new concept (developed by Viktor Tron [SWARM project lead]) that must be implemented in the SWARMDB roadmap is the representation of a *provable object traversal*. To incentivize long-term storage of manifest encoding embedded in the decentralized index, we require a guarantee of storage of manifest encoding analogous to the SWEAR (guarantee of storage of a chunk) that is also paired with the SWINDLE. When an index mapping is stored in the SWARM, a transaction receipt is stored just as with SWEAR. The transaction receipt can be mapped onto a "provable object traversal", a sequence of manifest encodings. The network protocol for delegated traversal and POT proof response is being developed in further detail.

Data on-boarders may specify a WOLK "bounty" for each record.   This bounty is kept in the manifest in our current POC.

3. Wolk Ecosystem

To support open data and content sharing Wolk will develop the Wolk Ecosystem:

- Wolk Inc., a for-profit Delaware corporation, and Wolk Foundation, a Swiss Foundation, promoting the development of a decentralized data marketplace
- Content/Data Suppliers, who onboard content + data and earn WOLK
- Content/Data Buyers, who pay WOLK to access content + data using the SWARMDB interfaces
- Storage and Bandwidth Providers, who run SWARMDB nodes and earn WOLK
- Decentralized Apps, who pay WOLK in exchange for decentralized NoSQL storage

The basic actors of this ecosystem are shown in Figure 1.

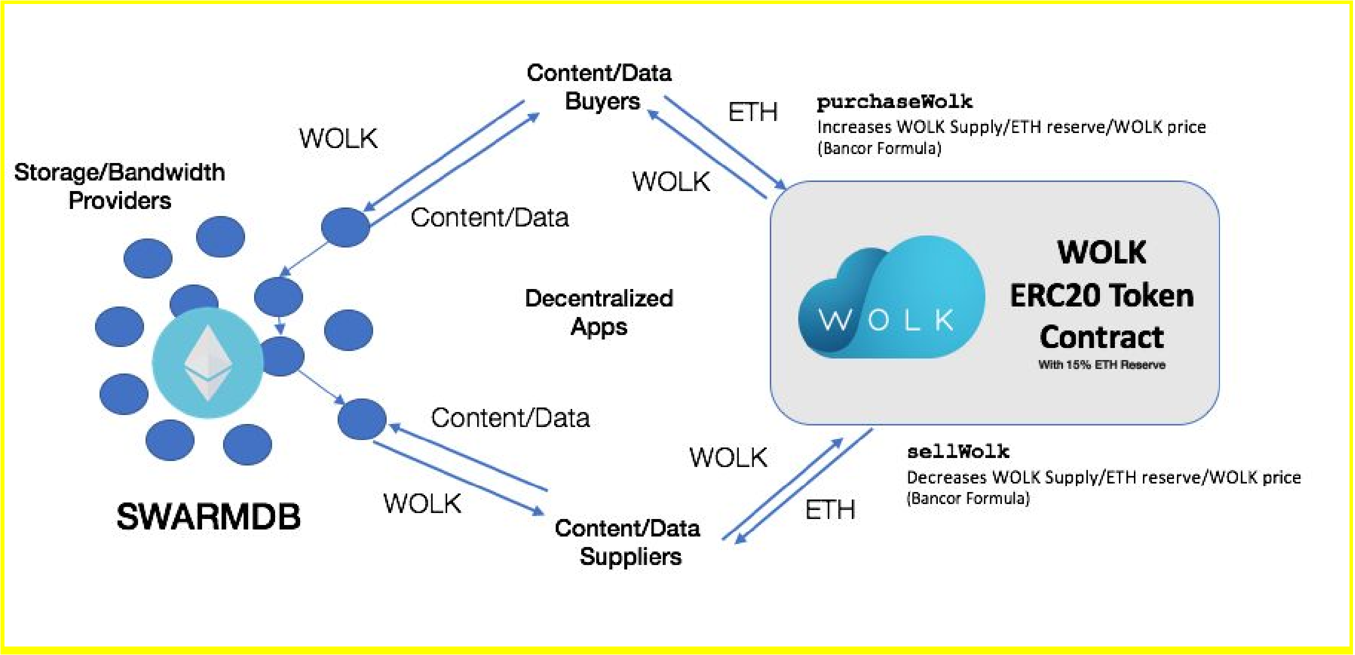
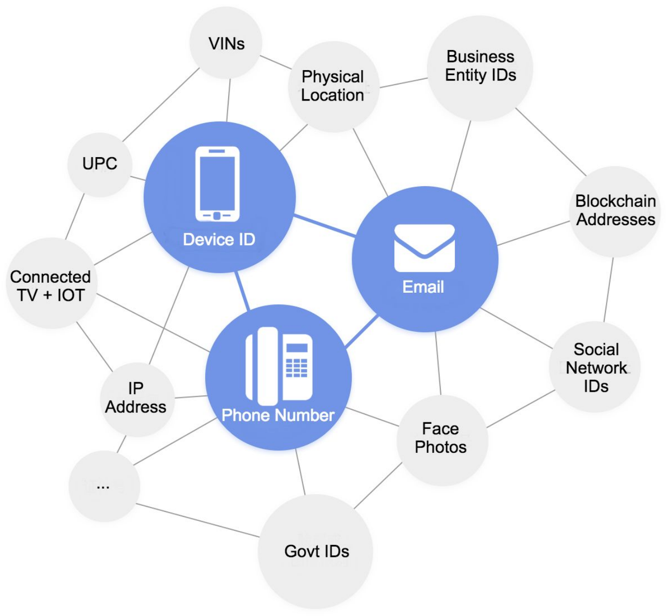

Content/Data Buyers
ETH
purchaseWolk
Increases WOLK Supply/ETH reserve/WOLK price (Bancor Formula)
WOLK
Storage/Bandwidth Providers
WOLK
Content/Data
Decentralized Apps
WOLK ERC20 Token Contract
With 15% ETH Reserve
Content/Data
WOLK
sellWolk
Decreases WOLK Supply/ETH reserve/WOLK price (Bancor Formula)
SWARMDB
WOLK
ETH
Content/Data Suppliers

Figure 1. Data + Monetary flow in Wolk Ecosystem.

The monetary flow is that "real money (ETH)" from decentralized applications and content/data buyers will flow into the network of storage + bandwidth providers and the content/data suppliers.

3.1 Content & Data Suppliers and Buyers

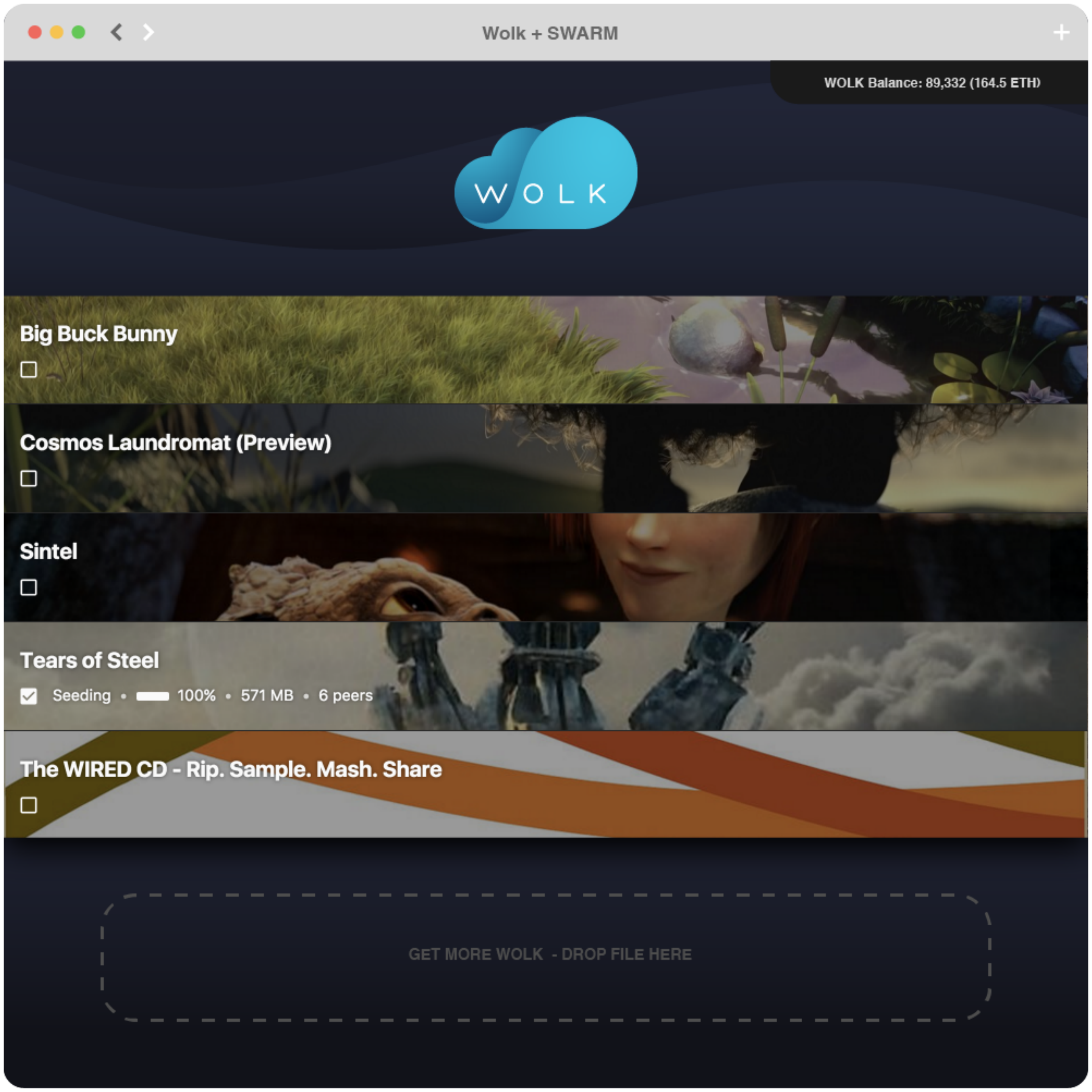
Wolk Inc. supports the onboarding of many ID spaces in SWARMDB about IDs in a wide variety of ID spaces, starting with emails, phone numbers and mobile device IDs:



For each ID space, standardized attributes will have own columns in public SWARMDB tables. Data suppliers will use one of SWARMDB interfaces to onboard their data. Content suppliers may do the same. As buyers access the data, sellers accumulate WOLK tokens. By allowing WOLK to be exchanged for ETH using a 15% ETH-reserve, Wolk provide data sellers a liquid market for their data, so long as ETH itself is liquid. In addition, the WOLK token is an ERC20 token, so with sufficient transaction volume and subject to the transfer restrictions described below, it will potentially be tradeable on a number of exchanges and supported by multiple cryptocurrency wallets, allowing data suppliers to monetize their WOLK token, although Wolk is not involved in these exchanges and cannot guarantee they will allow transfers of WOLK tokens.

3.2 Storage/Bandwidth Providers

To stimulate and support widespread adoption of a large WOLK-based SWARM, Wolk will release a consumer-friendly Electron-based implementation of a SWARM client based off an open source BitTorrent client (c.f. WebTorrent) that enable consumers to earn WOLK for content sharing and providing storage.


Wolk + SWARM
WOLK Balance: 89,332 (164.5 ETH)
WOLK
Big Buck Bunny
Cosmos Laundromat (Preview)
Sintel
Tears of Steel
Seeding • 100% • 571 MB • 6 peers
The WIRED CD - Rip. Sample. Mash. Share
GET MORE WOLK - DROP FILE HERE

Unlike BitTorrent clients, SWARM clients have the advantages of being highly censorship-resistant (because data cannot be be taken down, e.g. with DMCA notices), monetization oriented (because WOLK can earn storage providers ETH), and privacy conscious (because data of a file / record is not stored at any one specific node).

3.3 Decentralized Application Publishers

Decentralized applications users or developers need a large scale SWARM network. Wolk discussing with early beta candidates who need to fully decentralize their applications with SWARMDB.

3.4 Wolk Team

Wolk Inc. was created by professionals in mobile advertising that saw a need for decentralized data protocols. The Wolk Core Team consists of:

- Sourabh Niyogi, 45, Wolk CEO
- Sonia Gonzalez, 38, Wolk Inc GM
- Harish Thimmappa, 33, Wolk SVP Revenue
- Rodney Witcher, 36, Wolk VP Business Development
- Michael Chung, Wolk Product Manager and Protocol Developer
- Mayumi Matsumoto, Wolk Data Scientist and Protocol Designer
- Alina Chu, Wolk Data Scientist & Quantitative Analyst
- Nitin Chauhan, Wolk Data Engineer
- Bruce Han, Wolk Software Engineer
- Luvsanbyamba Buyankhuu, Wolk Data Engineer
- Scott Salin, Wolk Business Development

For more complete bios of our team, please visit our site team page here: https://wolk.com/app/team

Based on extensive experience and expertise in, among other things, implementing in-app advertising campaigns, modelling mobile deviceIDs and developing real-time bidding algorithms for the purpose of bidding on data exchanges, Wolk's principals and employees have developed first-hand knowledge of the relative dominance of Google and Facebook in mobile advertising, the need for real data solutions from existing vendors, and how digital marketing can be done more efficiently with the benefit of quality data at scale.

Wolk will host SWARM nodes alongside and support guardianship and ensure storage goals of (2) are being met. Successful large scale deployment will consist of:

- Less than 10^{-6} loss for low scale chunk data storage
- Demonstration of SWEAR incentive structure working with court system in practice
- Resistance to data loss

Wolk Inc. aims to develop and promote the Wolk Protocol starting with the *advertising* data ecosystem.

Wolk Inc is charged to:

- develop relationships with data buyers, who must be supported in using SWARMDB to obtain data
- develop and enhance SWARMDB for data suppliers to onboard data onto WOLK, and deliver increasing data quality for data buyers

The Wolk Foundation provides financial support for Wolk Inc.

Wolk's Roadmap anticipates milestones in multiple related threads:

1. SWARMDB development: permissioning development (proxy re-encryption, decentralized key management services), node incentive research, blockchain connectivity with very high throughput / low latency mechanism (c.f. Plasma).

2. data development -- Adding ID spaces and events concerning these ID spaces are natural to evolve
3. storage

4. WOLK Token Generation Event

4.1 Summary

Our goal is to distribute a minimum of 50MM WOLK and maximum of 200MM WOLK tokens in a "Token Generation Event" from September 18, 2017 (block # 4289200) until around October 17, 2017 (block # 4370000) in the following distributional pattern:

- Token Generating Event: Minimum: 50MM WOLK [around 42.5K ETH]
  Maximum: 200MM WOLK [around 188.8K ETH]
- Token Generation Event Start Date/Time: Block # 4289200 [September 18, 2017 23:00 GMT]
- Token Generation Event End Date/Time: Block # 4370000 [est. around October 17, 2017]

Token contract address: **0x728781E75735dc0962Df3a51d7Ef47E798A7107E [ENS: wolktoken.eth]**

Token launch completion: Token launch will end when either the maximum number of WOLK is raised (200MM WOLK) or the Targeted End Date/Time and the minimum number of WOLK has not been achieved (50MM WOLK). If less than the minimum WOLK are raised, ETH will be refunded to the sending addresses.

U.S. purchasers and non-U.S. purchasers may buy WOLK through the Wolk website. These WOLK will be issued by Wolk Inc. Non-U.S. purchasers may also buy WOLK through Bitcoin Suisse. These WOLK will be issued by the Wolk Foundation.

Wolk adopts a KYC practice for all WOLK Token Generation Participants. Participants must e-sign a purchase agreement on wolk.com. Up to a 15% discount will be granted based on the number of tokens issued at the time ETH (or USD) is received in accordance with the following schedule:

# of Tokens Issued At Time ETH Funds Received	Approximate % Discount / Pricing
0 to 49,999,999	15.0% - 1,176 WOLK per ETH (.000850 ETH/WOLK)
50,000,000 to 59,999,999	12.5% - 1,143 WOLK per ETH (.000875 ETH/WOLK)
60,000,000 to 69,999,999	10.0% - 1,111 WOLK per ETH (.000900 ETH/WOLK)
70,000,000 to 79,999,999	7.5% - 1,081 WOLK per ETH (.000925 ETH/WOLK)
80,000,000 to 89,999,999	5.0% - 1,052 WOLK per ETH (.000950 ETH/WOLK)
90,000,000 to 99,999,999	2.5% - 1,023 WOLK per ETH (.000975 ETH/WOLK)
100,000,000 to 199,999,999	0% - 1,000.00 WOLK per ETH (.00100 ETH/WOLK)

The 200MM WOLK is split into two separate allocations: 150MM WOLK for ETH-based purchases sent to the Wolk Token contract and 50MM WOLK for a small number of USD-based purchases sent to

Wolk's US-based bank account. USD-based purchases are finalized at the end of the Token Generation Event.

4.2 Wolk Token Economics

After the Token Launch, data buyers and suppliers may trade tokens freely on exchanges that allow such trades, subject to compliance with the restrictions on transfers described below, but may also use 2 smart contract functions - "purchaseWolk" and "sellWolk" - to exchange ETH for WOLK and WOLK for ETH freely, compensated from an ETH reserve in the smart contract. The implementation of purchaseWolk and sellWolk utilizes the "Bancor" formulas of calculatePurchaseReturn and calculateSaleReturn (see http://bancor.network white paper) where:

- data buyers will be able to buy WOLK by sending ETH to the "purchaseWolk" function of the WOLK Token Contract, which will use the ETH:WOLK exchange rate internally to the contract along with the calculatePurchaseReturn formula to determine the amount of WOLK to transfer to the buyer. The ETH contributed goes into a "ETH reserve" and the prevailing ETH:WOLK exchange rate is automatically updated. Buyers who obtain WOLK then use SWARMDB interfaces to obtain data from SWARM;
- data sellers will be able to sell WOLK by interacting with the "sellWolk" function of the WOLK Token Contract, specifying the amount of WOLK they wish to sell in exchange for ETH. Using the ETH:WOLK exchange rate internal to the contract as well as the calculateSaleReturn formula, ETH is reduced in the reserve and the ETH:WOLK exchange rate is automatically updated.

The total number of tokens therefore can increase from data buyers calling purchaseWolk and can decrease from data sellers calling sellWolk, driving WOLK's value up (via purchaseWolk) and down (via sellWolk) programmatically. Example 1 shows the token economics of how contributors can turn ETH to WOLK via purchaseWolk and convert WOLK back to ETH via sellWolk after the Token Generation Event is completed.

The actual price of WOLK is calculated as follows:

- R - Reserved ETH Balance in WOLK Token Contract
- S – Current Total Supply of WOLK Tokens
- F - Reserve Ratio, which is fixed at 15%

T = WOLK received in exchange for ETH E sent to purchaseWolk, given R, S and F

$$T = S\,((1 + \tfrac{E}{R})^{F} - 1)$$

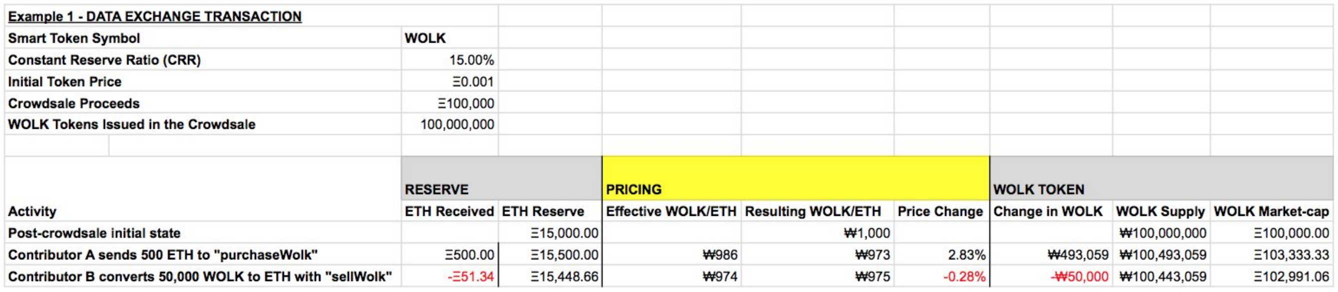
E = ETH received in exchange for T WOLK in sellWolk, given R, S and F

$$E = R\,(1 - \sqrt[F]{1 - \tfrac{T}{S}})$$

The derivations of the Bancor formulas are found here.

Example #1: WOLK Token Transactions

For this example, the Token Generation Event (TGE) has realized ETH 100,000 with 100,000,000



Example 1 - DATA EXCHANGE TRANSACTION								
Smart Token Symbol	WOLK							
Constant Reserve Ratio (CRR)	15.00%							
Initial Token Price	Ξ0.001							
Crowdsale Proceeds	Ξ100,000							
WOLK Tokens Issued in the Crowdsale	100,000,000							
	RESERVE		PRICING			WOLK TOKEN		
Activity	ETH Received	ETH Reserve	Effective WOLK/ETH	Resulting WOLK/ETH	Price Change	Change in WOLK	WOLK Supply	WOLK Market-cap
Post-crowdsale initial state		Ξ15,000.00		₩1,000			₩100,000,000	Ξ100,000.00
Contributor A sends 500 ETH to "purchaseWolk"	Ξ500.00	Ξ15,500.00	₩986	₩973	2.83%	₩493,059	₩100,493,059	Ξ103,333.33
Contributor B converts 50,000 WOLK to ETH with "sellWolk"	-Ξ51.34	Ξ15,448.66	₩974	₩975	-0.28%	-₩50,000	₩100,443,059	Ξ102,991.06

WOLK generated.   The ratio WOLK:ETH was 1000:1 and 15% ETH generated (ETH 15,000) is kept as a reserve.   Purchasing and exchanging WOLK through the reserve becomes possible as soon as the Token Generation Event is completed. The opening price is the TGE price, in this example 1 ETH for the 1,000 WOLK.

1. Contributor A converts ETH (Ξ) 500 into WOLK (493,059) by sending 500 ETH to "purchaseWolk". This 500 ETH is added to the ETH reserve (increasing to 15,500 ETH), and the Buyer receives 493,059 newly minted WOLK. This results in a WOLK price increase of 2.83% as ETH reserves go up.   The ETH reserve always remains 15% of the WOLK market cap, and market cap increases to ETH (Ξ) 103,333.33
2. Contributor B converts WOLK 50,000 to receive ETH (Ξ) 51.34. The ETH Reserve goes down by 51.34 ETH, resulting in a WOLK price decrease of -0.28% as the ETH reserves go down.   The token supply goes down by 50,000 WOLK, and the WOLK price decreases respectively. Market cap reduces to ETH (Ξ) 102,991.06

   Link to Google Sheets

After this "Token Generation Event", 15% of ETH will be kept in reserve to support the liquidity of WOLK:

1. WOLK Token holders may exchange WOLK for ETH and drive WOLK prices down [exactly as much as necessary to maintain the 15% ETH reserve]
2. ETH holders may exchange ETH for WOLK and drive WOLK prices up [again, exactly as much as necessary to maintain the 15% ETH reserve]

In addition, WOLK owners may sell their WOLK to data buyers or others using any exchanges listing WOLK, subject to restrictions on transferability described below (Wolk is not involved in these exchanges and cannot guarantee they will facilitate exchange of WOLK).

WOLK Buyers exchange ETH for WOLK and WOLK Sellers exchange WOLK for ETH via the WOLK Token Contract Functions purchaseWolk and sellWolk and can freely exchange WOLK on crypto exchanges.   An internal exchange rate is updated by purchaseWolk, sellWolk and with the goal of maintaining, as much as possible, a fixed 15% reserve following the Bancor formula.

The value of all WOLK tokens (and thus typically, the price of the WOLK token), can be proportional to the demand for all the data stored in the SWARM. Wolk Inc. has designed the WOLK Smart Token Contract to make exchanging WOLK to ETH and ETH to WOLK easy - using a 15% ETH - backed reserve (detailed later). SWARMDB is designed make it easy for data buyers to obtain data from SWARM and for data sellers to onboard data onto SWARM.

4.3 Restrictions on Transfers

The Wolk Tokens are being offered in reliance upon exemptions from registration under the Securities Act of 1933 ("**Securities Act**"). Therefore, unless the WOLK are used in commercial transactions using the WOLK protocols, WOLK may not be transferred within the United States or to a "U.S. person" unless such transfer is made to an "accredited investor," in compliance with applicable securities laws, and may only be transferred in a transaction outside the United States to non-U.S. persons, unless and until Wolk reasonably determines and notifies holders that the WOLK Tokens are not securities and freely tradeable. Any transfer made in violation of these provisions will be void.

Based on anticipated use of the WOLK and the Wolk protocols, Wolk believes that over time, the Tokens will reasonably be treated as non-securities for purposes of U.S. law, at which point Wolk will notify holders of the finding and that the WOLK Tokens are freely tradeable. There is no guarantee that this will occur.

4.4 Use of Proceeds

ETH raised in the Token Generation Event will be distributed as follows:

- 15% - supports liquidity of WOLK directly via purchaseWolk, sellWolk functions directly in the token contract, which increase this reserve programmatically
- 40% - supports Wolk Inc in developing SWARMDB and the Wolk Ecosystem and costs around marketing, business development, operations, etc.
- 45% - will support a "Wolk Ecosystem Development Fund," which will support content + data suppliers onboarding by Wolk Inc.

4.5 Wolk Inc Sustainability / Follow on Offerings

Wolk Inc is the service company charged with developing the content + data ecosystem. Wolk Inc sustainability is derived from:

- the WOLK earned from hosting SWARMDB nodes.
- the 40% of ETH from the Token Generation Event (see 5.4 above)

4.6 WOLK Token Generation Event FAQs

What does WOLK represent?

WOLK represents a decentralized virtual currency represented in specialized Ethereum smart contracts following an "ERC20" standard. WOLK enables participants to interact with SWARMDB. Data is obtained with WOLK as a payment mechanism in SWARMDB. Content + data suppliers and storage + bandwidth providers are compensated in WOLK for their highly valuable content, data, storage and bandwidth.

Are you distributing WOLK Tokens in the Token Generation Event?

Yes. Wolk Inc. and the Wolk Foundation are offering to distribute a minimum of 50MM WOLK and a maximum of 200MM WOLK in aggregate.

How do I obtain WOLK in the Token Generation Event?

To obtain WOLK from the Wolk web site, you must e-sign a Purchase Agreement and send ETH to the Wolk Token Contract from your Ethereum Wallet. To obtain WOLK via Bitcoin Suisse, you must register with Bitcoin Suisse and and provide your Ethereum wallet address to Bitcoin Suisse. Other cryptocurrencies must be exchanged for ETH to obtain WOLK. Wolk is allowing for a small number of USD-based purchases, which will be finalized at the end of the Token Generation Event.

What is the value of WOLK? Is WOLK transferable?

During the Token Generation Event, the value of WOLK is proportional to the value of ETH in the fixed schedule shown in 5.1. Afterwards, buyers and sellers of WOLK can potentially exchange WOLK freely on cryptocurrency exchanges, subject to the restrictions described above. In addition, the 15% ETH reserve held in the WOLK Token Smart Contract should enable systematic exchange of WOLK for ETH and vice versa with publicly visible exchange rates.

WOLK is transferable. Subject to the restrictions described above, it can be sold to buyers who require WOLK to procure data, to suppliers and cryptocurrency speculators. All transactions conducted are verifiable and secured on the public Ethereum blockchain.

Does WOLK represent ownership of Wolk?

No. WOLK does not represent any ownership rights in Wolk Inc or the Wolk Foundation and does not represent participation in Wolk Inc. or the Wolk Foundation.

When will the token launch happen?

Wolk Token Offering will be conducted between September 10, 2017 and October 17, 2017 in two stages:

Stage 1 Pre-Sale: September 10 (block # 426050) – September 18, 2017 (block # 4289200)

Stage 2 Main Token Generation Event: September 18 (block # 4289200) – October 17, 2017 (block #4370000)

with precise block numbers decided based on Ethereum mining rates. The purpose of the two stage token launch is to accommodate KYC practices of different coin demand aggregation platforms. Announcement of WOLK availability on each coin platform will be announced before the week of September 10, 2017.

As of this writing, over $10MM has been committed by registered pre-sale participants. There is no guarantee that all these committed purchasers will actually buy WOLK.

How will Wolk store ETH?

Wolk will use the standard Ethereum multisig wallet to store ETH.

For additional information, see our Term Sheet on WOLK and our Token Generation Event.

4.7 Communications

The primary communications vehicle for Wolk is at https://wolk.com -- no other communications vehicle is considered official at this time.

Wolk maintains a Twitter account at https://twitter.com/WolkInc and encourages Twitter users to follow Wolk **@WolkInc**.

4.8 Open Source

The WOLK Token Contract is available on https://github.com/wolktoken/token and has been posted at wolktoken.eth. All code for SWARMDB will be made open source.

5. Risks and Risk Mitigation

Wolk advises on the following risks:

I. Insufficient Data Supply or Demand. While Wolk is optimistic that decreasing CPMs and reduced revenue from competition Google and Facebook motivate data suppliers to seek new monetization options such as Wolk, the market for data monetization may be slower to develop given uncertainty on revenue potential. While Wolk has significant interest from mobile demand side platforms in attributes keyed in by mobile device IDs and is in testing, Wolk has validated other ID spaces. It is anticipated that other identifiers will be at similar demand levels but this has not been validated in the same way as mobile device IDs.

II. Insufficient liquidity for WOLK. WOLK is a specialty token for a much smaller B2B community of data suppliers and data buyers. A smaller number of participants implies lower volumes of WOLK transaction, which may result in higher volatility of WOLK.

III. Privacy and local regulations may pose unforeseen limits on Wolk. While Wolk will abide by industry standard privacy principles and control measures, many local markets operate with different principles and the restrictions may pose limits on what can be done in decentralized storage or on the blockchain structure.

IV. Data copying and fraud protection measures are being actively developed. Data buyers may copy data, reducing the ability for data suppliers to fully monetize their data because of the actions of fraudulent actors. Wolk has developed basic countermeasures for bad actors but there can be no assurances that the measures will be fully complete.

V. Transaction throughput may be insufficient to support extraordinarily high SWARMDB demand. However, but it is possible that higher throughputs will be necessary and significant investments will need to be made to improve Ethereum SWARM technology.

VI. Currently, WOLK is an Ethereum-backed token. Should further developments in the cryptocurrency ecosystem enable the handling of higher throughput, lower latency software architecture, Wolk wishes to remain nimble enough to transition over to better blockchain technology.

Wolk will use all available resources to reduce risks but outside of data buyers and suppliers participating, only accredited investors and investors comfortable with the above openly stated risks should hold WOLK tokens. For additional information on risks associated with purchasing WOLK, see our Term Sheet on WOLK and our Token Generation Event and our WOLK Risk Disclosures.