



Apollo Foundation

SHARDING VERIFICATION & AUDIT

07.11.2020

Made in Germany by Chainsulting.de



Table of contents

1. Disclaimer.....	3
2. About the Project and Company.....	4
2.1 Project Overview.....	5
3. Vulnerability & Risk Level	7
4. Auditing Strategy and Techniques Applied.....	8
4.1 Methodology	8
4.2 Codebase & Resources.....	9
5. Scope of Work & Results.....	9
6. Manual & Automated Vulnerability Test	10
7. Verification of Apollo Sharding.....	11
7.1 Sharding in detail	11
7.2 Description of Apollo Sharding	11
7.3 Technical deep dive.....	13
7.4 Apollo API related to Sharding.....	16
7.5 Verification of Sharding Process	17
8. Executive Summary.....	20



1. Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the codebase to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Apollo Foundation. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (04.11.2020)	Layout and details
0.5 (05.11.2020)	Automated Security Testing Manual Security Testing
0.8 (06.11.2020)	Sharding Verification
1.0 (07.11.2020)	Final document (Summary and Recommendation)

2. About the Project and Company

Company address:

Apollo Fintech Ltd.
Unit 1411, 14/Floor, Cosco Tower
183 Queen's Road Central
Hong Kong

Website: <https://aplfintech.com/>

GitHub: <https://github.com/ApolloFoundation/Apollo>

Twitter: <https://twitter.com/ApolloCurrency>

LinkedIn: <https://www.linkedin.com/company/apollo-foundation/>

Telegram: <https://t.me/apollocommunity>

Instagram: <https://www.instagram.com/apollocurrency/?hl=en>

Medium: <https://medium.com/@apollocurrency>

Explorer: <https://explorer.apollowallet.org/>

CMC: <https://coinmarketcap.com/de/currencies/apollo-currency/>

2.1 Project Overview

Utilizing a community of world-class developers, managers, marketers and researchers, the Apollo community, backed by the Apollo Foundation, has set out to accomplish the goal of making Apollo the most technologically advanced, feature-rich currency on the market. The Apollo Foundation understands the demands of a top tier cryptocurrency and they believe they can create a coin that will integrate everything necessary to replace the current currency options. As they develop and improve Apollo, the Apollo team will strive to develop features which have not previously used in cryptocurrency. The primary goal of the Apollo foundation is to create the first all-in-one cryptocurrency, innovating and incorporating every ability that could be beneficial in a digital currency, all in a single decentralized platform. The first major update, Olympus Protocol, puts mass adoption-proof privacy at Apollo's core. This is because the Apollo Foundation knows the ability for a user to buy, sell, trade and send in absolute secrecy is vital in an industry that could be moments away from intense regulation. The Apollo team is here because they believe the only person or entity that should be in control of your funds is you. The monetary system within Apollo allows users to instantly create currencies or tokens that can be traded privately on Apollo's decentralized exchange as well as freely on external exchanges.

Using the Apollo asset system, a user can issue tokens representing anything from public and private equity to real world commodities. Unlike other markets and cryptocurrencies, users will be able to trade these assets with 100% privacy. Encrypted messaging on the Apollo blockchain will allow any user to send and receive 100% private, untraceable messages and data files from one account to another. Utilizing the Olympus protocol, the user's IP address and the transaction carrying the message will be invisible.

The Apollo data cloud allows uploading to the blockchain for storage, retrieval and publishing of information. This gives a user the ability to upload a file to the blockchain, therefore receiving an irrefutable time stamp for that data. This time stamp could be helpful in establishing an absolute date for legal documents such as contracts and intellectual ownership.

Alias system in Apollo can be used to create a unique alias that acts as a representation for a specific set of text. This will encrypt the chosen text into the alias. This text could be your account number, a website, email address, etc.

The Apollo voting system allows users to create public or private polls. Polls can be used to direct and manage funds from an account, elect officials or simply to gauge public opinion.

Phased transactions are transactions set up to occur after a certain condition is met. This could be after another transaction is sent or received, after a passage of time or after a certain block number is hit.

Apollo supports multi-signature accounts allowing more than one user to control an account.

The Apollo decentralized marketplace allows users to buy, sell and trade physical and digital goods using the Apollo currency.

Accounts can be created in a way that allows group control. Utilizing the Apollo voting system, a group can be granted the ability (via initial ownership or ownership of a specified token or asset) to vote on the transactions that are spent on the account.

The Apollo authentication system allows users to authenticate an account using the blockchain to prove that they are the overseer of an account.

The account leasing option allows the user to lease the forging power of their account to another user. This allows for the creation of forging pools, increasing the chances of generating a block, as well as generating an income from transaction fees.

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the codebase functioning in a number of scenarios, or creates a risk that the codebase may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a codebase, or provides the opportunity to use a codebase in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the codebase in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the codebase and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and software developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the codebase.
 - ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
 - i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 - ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the codebase to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your codebase.



4.2 Codebase & Resources

Main Repo on Github:

<https://github.com/ApolloFoundation/Apollo>

Sharding:

<https://github.com/ApolloFoundation/Apollo/tree/master/apl-core/src/main/java/com/apollocurrency/aplwallet/apl/core/shard>

VoskCoin Apollo Sharding Verification Test:

<https://www.youtube.com/watch?v=pWge7m0SXHA&feature=youtu.be>

Tech paper:

<https://www.dropbox.com/s/fca6g9jq2dugmz9/Tech%20Document%202.0.pdf?dl=0>

5. Scope of Work & Results

The Apollo Team provided us with the files that needs to be tested. The scope of the audit is limited to the sharding of Apollo Blockchain, with the most code shared via github.

<https://github.com/ApolloFoundation/Apollo/tree/master/apl-core/src/main/java/com/apollocurrency/aplwallet/apl/core/shard>

The main goal of this audit was to verify that the Sharding works as expected and claimed by Apollo Foundation.

6. Manual & Automated Vulnerability Test

CRITICAL ISSUES

During the audit, Chainsulting's experts found **no Critical issues** in the codebase for Sharding.

HIGH ISSUES

During the audit, Chainsulting's experts found **no High issues** in the codebase for Sharding.

MEDIUM ISSUES

During the audit, Chainsulting's experts found **no Medium issues** in the codebase for Sharding.

LOW ISSUES

During the audit, Chainsulting's experts found **some Low issues** in the codebase for Sharding. Using Deep Dive and Code Warrior static analysis tool, shows some potential security vulnerabilities. However, these are just hints and would require further investigation to evaluate the real security risk. After our manual vulnerability check, we assumed that the overall security is not at risk regards the low issues. The results will be attached or shared via the audit report, as zip file.

7. Verification of Apollo Sharding

7.1 Sharding in detail

One of the biggest issues with current blockchain implementations in general is the number of transactions which can be processed per second. These numbers are quite low. Especially compared to established payment services like Visa.

The reason for that is that all nodes in a blockchain network do the same calculations. Though there may be thousands of nodes, they all together are not faster than a single node. In contrary they are even slower for the network overhead which is required to make the blockchain secure.

The term shard originates from databases where the data is spread across several machines in order to spread the load even from a single database query.

Sharding addresses this issue by dividing the blockchain network into smaller shards which operate on their own set of data and transactions. In this way the processing speed profits from the number of nodes in the network.

7.2 Description of Apollo Sharding

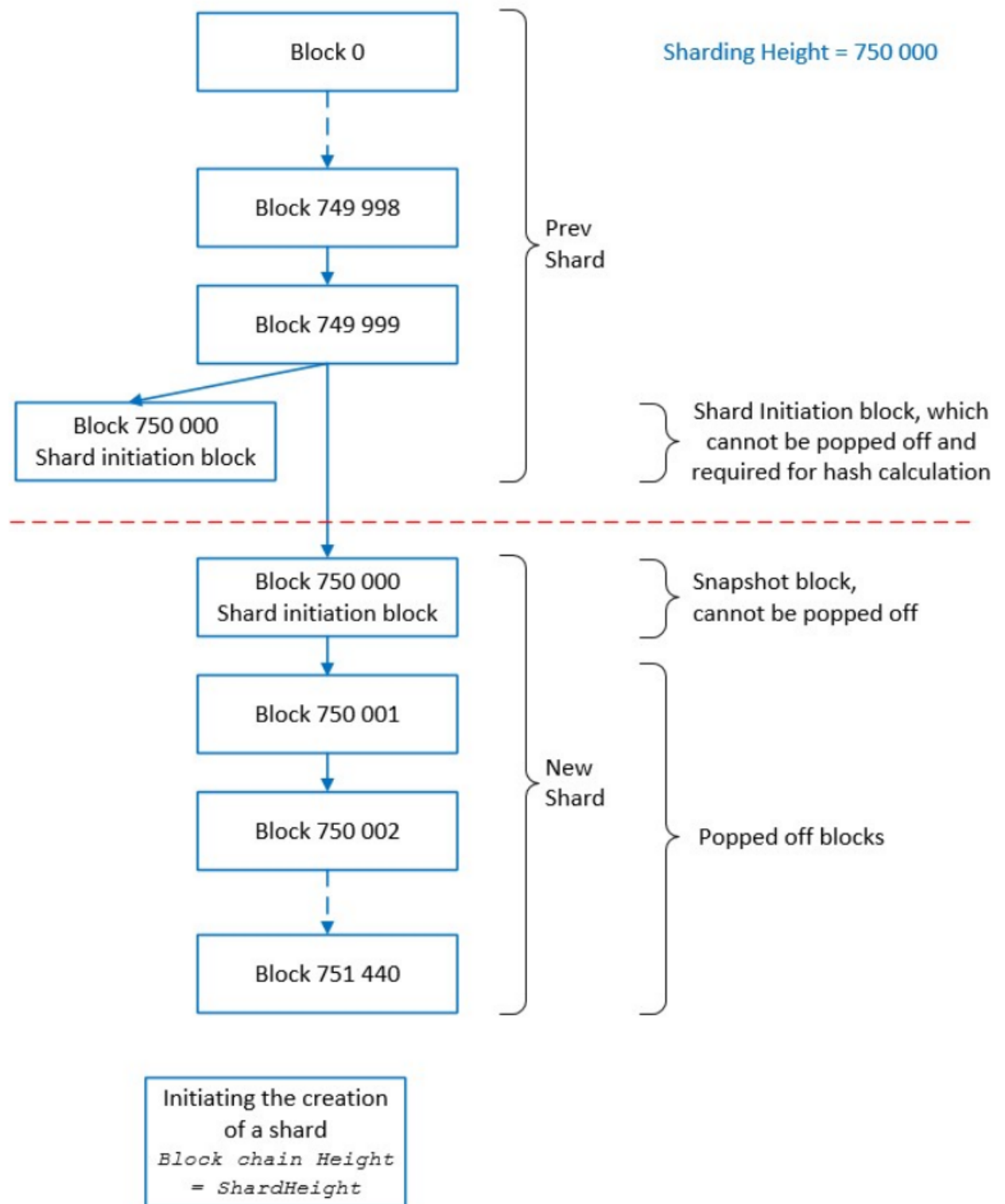
Apollo uses the term sharding in a different way. A shard in Apollo is like a super block which contains normal blocks. After a certain number of blocks these blocks are hashed and this hash is written in a new block on the blockchain. This ensures the provable correctness of the shard via the blockchain itself.

The point when a new shard is calculated by this formula.

$$ShardHeight = \frac{CurrentHeight - MaxRollback}{ShardFrequency}$$

The ShardFrequency is the number of blocks in a shard. CurrentHeight is the number of blocks in the blockchain. The subtraction of MaxRollback ensures that a shard is not created before all blocks - which will be inside the shard - can not be rolled back anymore.

The idea behind this is, that a node does not need to download the entire blockchain in order to participate in the network. It needs to download the most recent completed shard and the following blocks only. The correctness of the downloaded shard would be ensured by the hash over its whole. Therefore it is not necessary to review the chain of blocks inside of it.



7.3 Technical deep dive

In order to run an Apollo node its best to start on their github page: "<https://github.com/ApolloFoundation/Apollo>". There you can find a brief documentation with helpful advice. I suggest to clone the repository first with the following command

```
$ git clone https://github.com/ApolloFoundation/Apollo.git
```

Enter the Apollo folder and check the environment.

```
$ cd Apollo
$ ./mvnw -v
```

You should get an output like this

```
chainsulting@nodezero:~/test/Apollo$ ./mvnw -v
Apache Maven 3.6.2 (40f52333136460af0dc0d7232c0dc0bcf0d9e117; 2019-08-27T17:06:16+02:00)
Maven home: /home/chainsulting/.m2/wrapper/dists/apache-maven-3.6.2-bin/795eh28tki48bv3l67maojf0ra/apache-maven-3.6.2
Java version: 11.0.9, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: de_DE, platform encoding: UTF-8
OS name: "linux", version: "5.4.0-52-generic", arch: "amd64", family: "unix"
```

This means that your environment should be properly set up and you can build the project.

```
$ ./build.sh
```

If you are behind a firewall using NAT, you should forward the port 47874. In addition, Apollo had issues to get the external IP. The solution was to write it in the configuration file.

```
$ mkdir -p ~/.apl-blockchain/conf
$ cp doc/apl-blockchain.properties ~/.apl-blockchain/conf
$ vim ~/.apl-blockchain/conf/apl-blockchain.properties
```

There you can add your external IP to the key "apl.myAddress".

Now you can start the node.

If your Apollo files are on a drive, where space is sparse, you can use the parameter `--db-dir <path>` to choose a different location for the database. The database can grow rapidly on start up, however it will shrink later on.

The parameter “--no-shard-import false” will start the node for minimal disk usage. Only the latest completed shard and the following blocks will be downloaded.

```
$ ./bin/apl-run.sh --db-dir /media/chainsulting/apollo2 -s -d 3 --net 0 --no-shard-import false
```

```
chainsulting@nodezero:~/test/Apollo$ ./bin/apl-run.sh --db-dir /media/chainsulting/apollo2 -s -d 3 --net 0 --no-shard-import false
=== Apollo === Apollo wallet installed in directory: /home/chainsulting/test/Apollo
=== Apollo === Apollo wallet libraries installed in directory: /home/chainsulting/test/Apollo/apl-exec/target/lib
=== Apollo === Apollo versions is: 1.44.3
=== Apollo === Apollo main jar path: /home/chainsulting/test/Apollo/apl-exec/target/apl-exec-1.44.3.jar
=== Apollo === Using java at path: /usr/lib/jvm/java-11-openjdk-amd64/bin; Version is: 11; Java is OK.
Initializing Apollo
javafx not supported
Old chains: null
New chains: {a2e9b946-290b-48b6-9985-dc2e5a5860a1=Chain{chainId=a2e9b946-290b-48b6-9985-dc2e5a5860a1, active=false, defaultPeers=[0.80, 51.15.100.44, 51.15.233.93], blacklistedPeers=[], name='Apollo public testnet', description='Apollo main test chain, running', sLocation='data/genesisAccounts-testnet.json', featuresHeightRequirement='null', blockchainProperties={0=BlockchainProperties{heightLimit=53, maxBalance=30000000000, shardingSettings=ShardingSettings{, enabled=false, frequency=500000, digestAlgorithm='SHA-256'}}
```

The database is either in apl-blockchain-db in the Apollo folder, or if you have used --db-dir in the folder you have chosen.

In order to find anything of interest there, you will have to wait until the node is synchronized with the network. After this has happen you will find the following files in the folders.

```
chainsulting@nodezero:/media/chainsulting/apollo2$ ll -h
insgesamt 451M
drwxrwxrwx 1 chainsulting chainsulting 416 Nov 6 14:49 ./
drwxrwxrwx 1 chainsulting chainsulting 12K Nov 6 12:09 ../
drwxrwxrwx 1 chainsulting chainsulting 28K Nov 6 14:49 apl-blockchain/
drwxrwxrwx 1 chainsulting chainsulting 8,0K Nov 6 10:33 apl-blockchain-export-data/
-rwxrwxrwx 1 chainsulting chainsulting 451M Nov 6 14:51 apl-blockchain.mv.db*
-rwxrwxrwx 1 chainsulting chainsulting 2,1K Nov 6 10:41 apl-blockchain.trace.db*
chainsulting@nodezero:/media/chainsulting/apollo2$ ll -h apl-blockchain-export-data/
insgesamt 126M
drwxrwxrwx 1 chainsulting chainsulting 8,0K Nov 6 10:33 ./
drwxrwxrwx 1 chainsulting chainsulting 416 Nov 6 14:54 ../
-rwxrwxrwx 1 chainsulting chainsulting 126M Nov 6 10:12 apl-blockchain-shard-6-chain-b5d7b697-f359-4ce5-a619-fa34b6fb01a5.zip*
```

As you can see, inside the folder apl-blockchain-export-data is a zip file of the latest complete shard.

If you extract it, you will see that the data of the blocks is stored in several csv files.

```
chainsulting@nodezero:/media/chainsulting/shard$ ll -h
insgesamt 260M
drwxrwxrwx 1 chainsulting chainsulting 176 Nov 6 15:03 ./
drwxrwxrwx 1 chainsulting chainsulting 12K Nov 6 12:09 ../
-rwxrwxrwx 1 chainsulting chainsulting 8,6K Jan 1 1970 account_asset.csv*
-rwxrwxrwx 1 chainsulting chainsulting 489 Jan 1 1970 account_control_phasing.csv*
-rwxrwxrwx 1 chainsulting chainsulting 9,7M Jan 1 1970 account.csv*
-rwxrwxrwx 1 chainsulting chainsulting 866K Jan 1 1970 account_currency.csv*
-rwxrwxrwx 1 chainsulting chainsulting 3,7K Jan 1 1970 account_guaranteed_balance.csv*
-rwxrwxrwx 1 chainsulting chainsulting 580K Jan 1 1970 account_info.csv*
-rwxrwxrwx 1 chainsulting chainsulting 2,6K Jan 1 1970 account_lease.csv*
-rwxrwxrwx 1 chainsulting chainsulting 2,5K Jan 1 1970 account_property.csv*
-rwxrwxrwx 1 chainsulting chainsulting 16M Jan 1 1970 alias.csv*
-rwxrwxrwx 1 chainsulting chainsulting 313K Jan 1 1970 alias_offer.csv*
-rwxrwxrwx 1 chainsulting chainsulting 16K Jan 1 1970 ask_order.csv*
-rwxrwxrwx 1 chainsulting chainsulting 14K Jan 1 1970 asset.csv*
-rwxrwxrwx 1 chainsulting chainsulting 535 Jan 1 1970 asset_delete.csv*
-rwxrwxrwx 1 chainsulting chainsulting 377 Jan 1 1970 asset_dividend.csv*
-rwxrwxrwx 1 chainsulting chainsulting 9,9K Jan 1 1970 asset_transfer.csv*
-rwxrwxrwx 1 chainsulting chainsulting 2,8K Jan 1 1970 bid_order.csv*
-rwxrwxrwx 1 chainsulting chainsulting 751 Jan 1 1970 block.csv*
-rwxrwxrwx 1 chainsulting chainsulting 162M Jan 1 1970 block_index.csv*
-rwxrwxrwx 1 chainsulting chainsulting 668 Jan 1 1970 buy_offer.csv*
-rwxrwxrwx 1 chainsulting chainsulting 19K Jan 1 1970 currency.csv*
-rwxrwxrwx 1 chainsulting chainsulting 125 Jan 1 1970 currency_founder.csv*
-rwxrwxrwx 1 chainsulting chainsulting 124 Jan 1 1970 currency_mint.csv*
-rwxrwxrwx 1 chainsulting chainsulting 492 Jan 1 1970 currency_supply.csv*
-rwxrwxrwx 1 chainsulting chainsulting 1,9M Jan 1 1970 currency_transfer.csv*
-rwxrwxrwx 1 chainsulting chainsulting 13K Jan 1 1970 dex_contract.csv*
-rwxrwxrwx 1 chainsulting chainsulting 89K Jan 1 1970 dex_offer.csv*
-rwxrwxrwx 1 chainsulting chainsulting 16K Jan 1 1970 exchange.csv*
-rwxrwxrwx 1 chainsulting chainsulting 17K Jan 1 1970 exchange_request.csv*
-rwxrwxrwx 1 chainsulting chainsulting 325K Jan 1 1970 goods.csv*
-rwxrwxrwx 1 chainsulting chainsulting 1,1K Jan 1 1970 phasing_approval_tx.csv*
-rwxrwxrwx 1 chainsulting chainsulting 2,4K Jan 1 1970 phasing_poll_result.csv*
-rwxrwxrwx 1 chainsulting chainsulting 29K Jan 1 1970 poll.csv*
-rwxrwxrwx 1 chainsulting chainsulting 11K Jan 1 1970 poll_result.csv*
-rwxrwxrwx 1 chainsulting chainsulting 15M Jan 1 1970 public_key.csv*
-rwxrwxrwx 1 chainsulting chainsulting 33K Jan 1 1970 purchase.csv*
-rwxrwxrwx 1 chainsulting chainsulting 437 Jan 1 1970 purchase_feedback.csv*
-rwxrwxrwx 1 chainsulting chainsulting 347 Jan 1 1970 purchase_public_feedback.csv*
-rwxrwxrwx 1 chainsulting chainsulting 170 Jan 1 1970 referenced_transaction.csv*
-rwxrwxrwx 1 chainsulting chainsulting 704 Jan 1 1970 sell_offer.csv*
-rwxrwxrwx 1 chainsulting chainsulting 1,4K Jan 1 1970 shard.csv*
-rwxrwxrwx 1 chainsulting chainsulting 20K Jan 1 1970 tag.csv*
-rwxrwxrwx 1 chainsulting chainsulting 94 Jan 1 1970 tagged_data_extend.csv*
-rwxrwxrwx 1 chainsulting chainsulting 1,5K Jan 1 1970 tagged_data_timestamp.csv*
-rwxrwxrwx 1 chainsulting chainsulting 13K Jan 1 1970 trade.csv*
-rwxrwxrwx 1 chainsulting chainsulting 664 Jan 1 1970 transaction.csv*
-rwxrwxrwx 1 chainsulting chainsulting 56M Jan 1 1970 transaction_shard_index.csv*
```

The configuration file “apl-blockchain.properties” has some options regarding sharding which are listed here

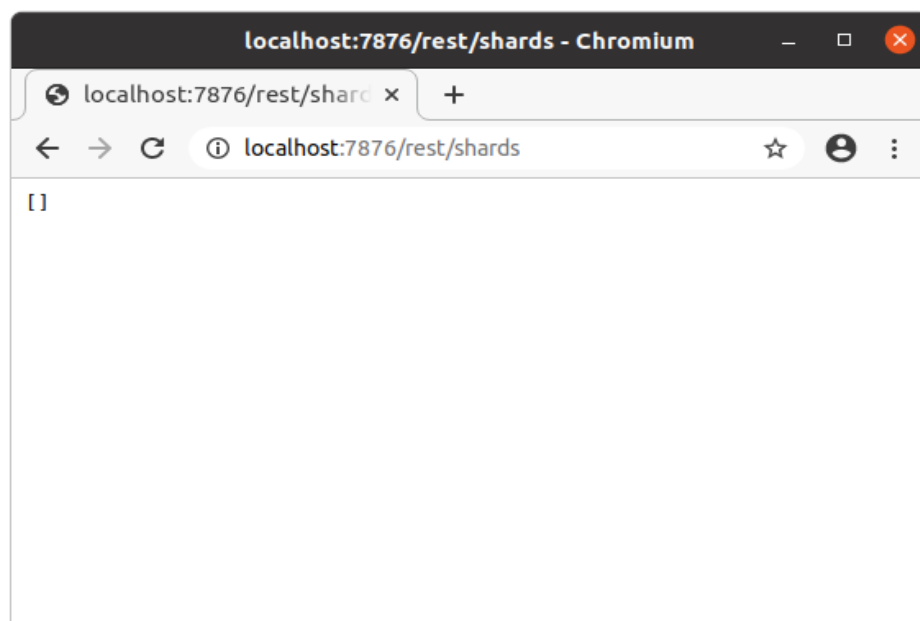
Key	Description	Type
apl.sharding.backupDb	Enables to backup the database before extracting a shard from it	Boolean
apl.noshardimport	Enables to download all blocks from the beginning instead of the latest complete shard only	Boolean
apl.noshardcreate	Enables no to split data into shards, but keep all data in one database	Boolean

7.4 Apollo API related to Sharding

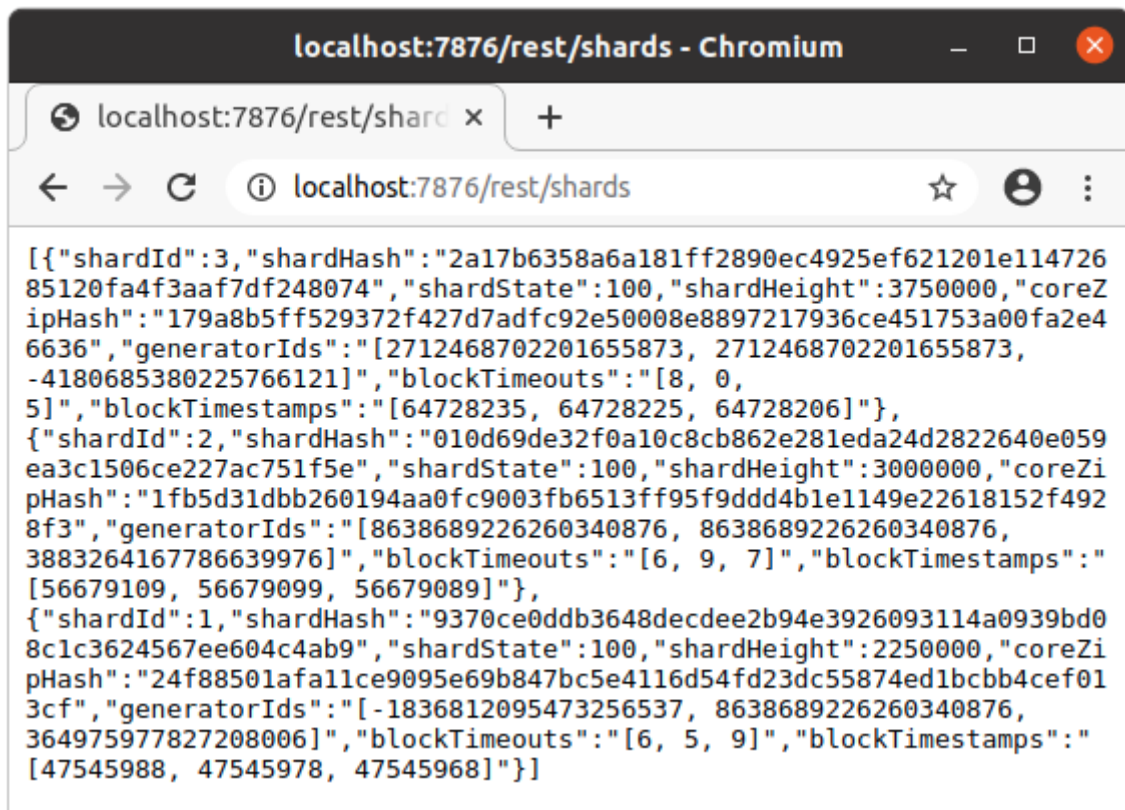
The Apollo API offers a call to show the downloaded shards

`http://localhost:7876/rest/shards`

The result is a json object. After I started the node like described above and letting it sync, I got an empty object.



However, I did some more testing and ran a node which shall download the entire blockchain. I did not sync it fully, but I got far enough to download some shards. This time the API showed some shards.



```

[{"shardId":3,"shardHash":"2a17b6358a6a181ff2890ec4925ef621201e11472685120fa4f3aaf7df248074","shardState":100,"shardHeight":3750000,"coreZipHash":"179a8b5ff529372f427d7adfc92e50008e8897217936ce451753a00fa2e46636","generatorIds":["2712468702201655873, 2712468702201655873, -4180685380225766121"],"blockTimeouts":["8, 0, 5"],"blockTimestamps":["64728235, 64728225, 64728206"]}, {"shardId":2,"shardHash":"010d69de32f0a10c8cb862e281eda24d2822640e059ea3c1506ce227ac751f5e","shardState":100,"shardHeight":3000000,"coreZipHash":"1fb5d31dbb260194aa0fc9003fb6513ff95f9ddd4b1e1149e22618152f4928f3","generatorIds":["8638689226260340876, 8638689226260340876, 3883264167786639976"],"blockTimeouts":["6, 9, 7"],"blockTimestamps":["56679109, 56679099, 56679089"]}, {"shardId":1,"shardHash":"9370ce0ddb3648decdee2b94e3926093114a0939bd08c1c3624567ee604c4ab9","shardState":100,"shardHeight":2250000,"coreZipHash":"24f88501afallce9095e69b847bc5e4116d54fd23dc55874ed1bcbb4cef013cf","generatorIds":["-1836812095473256537, 8638689226260340876, 364975977827208006"],"blockTimeouts":["6, 5, 9"],"blockTimestamps":["47545988, 47545978, 47545968"]}]]

```

7.5 Verification of Sharding Process

The sharding service is started in the class
“com.apollocurrency.aplwallet.apl.core.app.AplCore”

```




















264     ApiSplitFilter.isCoreReady = true;
265
266     PrunableArchiveMigrator migrator = CDI.current().select(PrunableArchiveMigrator.class).get();
267     migrator.migrate();
268     // start shard process recovery after initialization of all derived tables but before launching threads (blockchain
269     // downloading, transaction processing)
270     recoverSharding();
271
272     //start all background tasks
273     taskDispatchManager.dispatch();
274
275     try {
276         secureRandomInitThread.join(10000);
277     } catch (InterruptedException ignore) {}

```

The packages with the sharding code

- ▶  com.apollocurrency.aplwallet.apl.core.shard
- ▶  com.apollocurrency.aplwallet.apl.core.shard.commands
- ▶  com.apollocurrency.aplwallet.apl.core.shard.hash
- ▶  com.apollocurrency.aplwallet.apl.core.shard.helper
- ▶  com.apollocurrency.aplwallet.apl.core.shard.helper.csv
- ▶  com.apollocurrency.aplwallet.apl.core.shard.helper.jdbc
- ▶  com.apollocurrency.aplwallet.apl.core.shard.model
- ▶  com.apollocurrency.aplwallet.apl.core.shard.observer
- ▶  com.apollocurrency.aplwallet.apl.core.shard.observer.events
- ▶  com.apollocurrency.aplwallet.apl.core.shard.util

This is the main sharding package. The other packages are there to support it.

- ▼  com.apollocurrency.aplwallet.apl.core.shard
 -  BlockIndexService.java
 -  BlockIndexServiceImpl.java
 -  DbHotSwapConfig.java
 -  ExcludedTransactionDbIdExtractor.java
 -  MigrateState.java
 -  PrevBlockInfoExtractor.java
 -  PrunableArchiveMigrator.java
 -  PrunableArchiveMonitor.java
 -  ShardArchiveProcessingException.java
 -  ShardConstants.java
 -  ShardEngine.java
 -  ShardEngineImpl.java
 -  ShardImporter.java
 -  ShardManagement.java
 -  ShardMigrationExecutor.java
 -  ShardNameHelper.java
 -  ShardPrunableZipHashCalculator.java
 -  ShardService.java

The actual work is done by the class “com.apollocurrency.aplwallet.apl.core.shard.ShardEngineImpl”. Like creating the CSV files.

```
437 @Override
438 public MigrateState exportCsv(CommandParamInfo paramInfo) {
439     checkRequiredParameters(paramInfo);
440     long startTime = System.currentTimeMillis();
441     List<TableInfo> allTables = paramInfo.getTableInfoList();
442     log.debug("Starting EXPORT data from 'derived tables' + [{}], allTables.size());
443
444     ShardRecovery recovery = getOrCreateRecovery(CSV_EXPORT_STARTED);
445     if (recovery.getState().getValue() >= CSV_EXPORT_FINISHED.getValue()) {
446         // skip to next step
447         return state = CSV_EXPORT_FINISHED;
448     }
449     state = CSV_EXPORT_STARTED;
450     durableTaskUpdateByState(state, 17.0, "CSV exporting...");
451     try {
452         int pruningTime = trimDerivedTables(paramInfo.getSnapshotBlockHeight() + 1);
453         if (StringUtil.isBlank(recovery.getProcessedObject())) {
454             try (Stream<Path> files = Files.list(csvExporter.getDataExportPath())) {
455                 files
456                     .filter(p -> !Files.isDirectory(p) && p.toString().endsWith(CsvAbstractBase.CSV_FILE_EXTENSION))
457                     .forEach(FileUtils::deleteFileIfExistsQuietly);
```

Also the zipping and creating the hash code of the resulting zip file.

```
649 private void doZip(ShardRecovery recovery, String zipName, FilenameFilter fileFilter, BiConsumer<Shard, byte[]>
650 postCompressTask) {
651     if (AbstractHelper.isContain(recovery.getProcessedObject(), zipName)) {
652         log.debug("Skip already performed compression for {} ", zipName);
653     } else {
654         Path zipPath = dirProvider.getDataExportDir().resolve(zipName);
655         log.debug("Zip file name = '{}' will be searched/stored in '{}'", zipName, zipPath);
656         // delete if something left in previous run
657         boolean isRemoved = FileUtils.deleteFileIfExists(zipPath);
658         log.debug("Previous Zip in '{}' was '{}', zipName, isRemoved ? \"REMOVED\" : \"NOT FOUND\"");
659         // compute ZIP crc hash
660         ChunkedFileOps fops = zipComponent.compressAndHash(
661             zipPath.toAbsolutePath().toString(),
662             dirProvider.getDataExportDir().toAbsolutePath().toString(), null, fileFilter, false);
663         byte[] zipCrcHash = null;
664         if (fops != null && fops.isHashedOK()) {
665             zipCrcHash = fops.getFileHash();
666         }
667         //inform DownloadableFileManager
668         postCompressTask.accept(requireLastNotFinishedShard(), zipCrcHash);
669         updateRecovery(recovery, zipName);
670     }
671 }
```

8. Executive Summary

The codebase is written as simple as possible and not overloaded with unnecessary functions. Most of the codebase is well commented, these is greatly benefiting the review process, to fast understand how everything is supposed to work. The main goal of the audit was to verify the claims regarding the Sharding (see the Scope of work section). According to the code, the implementation of this functions are working and after our testing we can verify that both claims are valid. During the security audit of the Sharding function, no critical or high issues were found after the manual and automated security testing.