Cardano Audit Report

February 2018

This report is based on the following commits.

- `cardano-sl`: `535c36cf9496958e96aabf57bb875012060b3b34`
- `daedalus`: `d9afc5d4308b3b0f8f2467d3db8bb54c4746537e`

This is not a final report, but a snapshot of what has been reviewed thus far.

Notice

NOTE: This report is intended for public distribution.

This audit report should be considered preliminary work only. It is known to be incomplete. Crucial information for the project has not yet been reviewed. Some statements within may be incorrect. We will strive to improve on these areas in future iterations of the report.

Table of Contents

Disclaimer

THIS REPORT IS UNCORRECTED AND INCOMPLETE, IS BASED ON INCOMPLETE INFORMATION, AND IS PRESENTED WITHOUT WARRANTY OR GUARANTY OF ANY TYPE. This report is a copy of a work in progress. It lists the most salient concerns that have so far become apparent to FP Complete after a partial inspection of the engineering work. The inspection is ongoing, so further concerns are likely to arise. Corrections, such as the cancellation of incorrectly reported issues, may also arise. Therefore FP Complete advises against making any business decision or other decision on the basis of this report.

FP COMPLETE DOES NOT RECOMMEND FOR OR AGAINST THE USE OF ANY WORK OR SUPPLIER REFERENCED IN THIS REPORT. This report focuses on the technical implementation as provided by the project's contractors and subcontractors, based on information provided by them, and is not meant to assess the concept, mathematical validity, or business validity of the Cardano blockchain. This report does not make any assessment of the implementation or the cryptocurrency regarding financial viability, nor suitability for any purpose. While this assessment when complete might be described as an “audit,” no official standard exists for an audit of this nature. The word “audit” does not imply compliance with an accounting standard or other standard, and is used informally here.

FP Complete has not yet been provided complete and timely access to all aspects of the project and the engineering decision process underlying all of the work. Nor has FP Complete had direct access to most of the engineering personnel, who might shed light on some areas that caused concern, because they are busy. This report likely contains errors due to incomplete information as well as simple misunderstanding. This report may include references to problems that do not in fact exist. Meanwhile, the work referenced may or may not contain undetected or unreported problems. FP Complete has not had independent and unfettered access to all the relevant materials. Nor has a “whistleblower” or other process been provided such that any known problem could be reported and included herein.

Some technical decisions in the engineering work were made due to historic reasons, time constraints, budget constraints, or other constraints. Therefore the presence of a concern or “flag” in this report does not imply improper conduct or lack of skill by the implementer or manager or any party. NO ATTEMPT IS MADE OR IMPLIED TO JUDGE ANY PERSON, TEAM, COMPANY, OR OTHER PARTY.

Excluded content

Certain aspects of the project are outside of the scope of this audit report, though they are known to be relevant to project success. Future versions of this report may include coverage of these or additional topics. Topics known to be relevant but not covered include, but are not limited to:

- Devops
- IT Operations
- Project Management

In addition, some known issues may be excluded from a public report due to the security impact of premature public disclosure.

Legend

The following report uses a grading system that provides a subjective rating of the impact, significance and execution of the audited aspects. This is intended to assist the audience’s understanding.

Green Well executed in the project

Yellow Potential flaw, or process limitation which may lead to a flaw

Red Confirmed flaw

Summary

Title	Links	Status
Code		
Overuse of unvetted shell script code	Details IOHK Response	In progress
Relatively low number of lines of code for test suites	Details IOHK Response	In progress
The project uses the `error` function instead of correctly dealing with failure cases	Details IOHK Response	In progress
Using the non-best practice `Given` for configuration instead of `ReaderT`	Details IOHK Response	Acknowledged
Over-use of monad transformers	Details IOHK Response	Low priority
Some partial functions (e.g. `decodeUtf8`) from `text` are not hidden	Details IOHK Response	Minor issue
Use of partial functions in the project	Details IOHK Response	Minor issue
Custom prelude	Details	
Partial functions from base are not exposed	Details	
Many standard mistakes are eliminated by Universum	Details	
Code management		
Many bugs lack regression testing in resolution	Details IOHK Response	In progress
Daedalus does not pin a version of the `cardano-sl` codebase	Details IOHK Response	In progress
The `master` branch is not the default branch to clone on Github	Details IOHK Response	Will review later
The project uses git-flow	Details	
There exists a document that describes the git workflow	Details	
Continuous integration		
Opt-in nature of CI may lead to undiscovered test failures	Details IOHK Response	In progress
Haddock API documentation fails to build	Details IOHK Response	Reported Complete
The CI for master isn't all-green	Details IOHK Response	Reported Complete

Title	Links	Status
Buildkite logs are not public	Details IOHK Response	Blocked by vendor
Test coverage is not run in CI	Details IOHK Response	In backlog
Haddocks are not built in CI	Details IOHK Response	In backlog
The project uses Nix on CI	Details	
The project uses CI	Details	
Dependencies		
Duplicated solutions	Details IOHK Response	In progress
No evidence of vetting of dependencies	Details IOHK Response	In progress
The project uses relatively untested log-warper	Details IOHK Response	In progress
Blocks are inefficiently stored in individual files	Details IOHK Response	Acknowledged
Usage of non-transactional data storage layer	Details IOHK Response	Will review later
The project uses a non-standard variant of JSON	Details IOHK Response	Will not fix
Documentation		
The building/contribution guide does not work on all systems	Details IOHK Response	In progress
Some documents have no owner	Details IOHK Response	In progress
Links to internal documents in the public documentation	Details IOHK Response	In progress
There are runtime errors in the demo	Details IOHK Response	Acknowledged
There is no Readme for most packages, and no overview of packages	Details IOHK Response	Acknowledged
There is a public roadmap document	Details	
Each project management document has a 'purpose' section	Details	
Quality		
Insufficient variety in testing approaches	Details IOHK Response	Acknowledged
No evidence received of security-focused testing	Details IOHK Response	Acknowledged

Title	Links	Status
● **Static verification of code correctness**	Details	
● **Memory safety by default**	Details	
● **Well structured Byron Updates test plan**	Details	

Detailed flags: Code

	Public	Private	Total
Old flags	7	2	9
New flags	3	0	3
Total flags	10	2	12
Newly resolved flags	0	0	0
Red flags	0	1	1
Yellow flags	7	1	8
Green flags	3	0	3

Overuse of unvetted shell script code

The project contains a significant number of shell scripts:

```
cloc --include-lang='Bourne Shell, Bourne Again Shell' .
      79 text files.
      73 unique files.
      33 files ignored.

github.com/AlDanial/cloc v 1.74  T=0.11 s (426.6 files/s, 22536.3 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           co
-------------------------------------------------------------------------------
Bourne Shell                    46            425            309           16
-------------------------------------------------------------------------------
SUM:                            46            425            309           16
-------------------------------------------------------------------------------
```

These scripts are not shellcheck-clean, which may indicate correctness issues.

```
$ find . -type f -name '*.sh' -exec shellcheck {} \; | wc -l
1131
```

These shell scripts could introduce vulnerabilities in the worst case, and serious development overhead in the best case.

The following is a sample of some pull requests that relate to issues that stem from the usage of a shell script.

- PR 2086
- PR 2062
- PR 1927

First occurrence: December 2017

Status: In progress

IOHK Response

> It is being tracked by the following issue: DEVOPS-678: reduce amount of shell scripts and lint with Shellcheck. DevOps are cleaning up their scripts and this effort will be replicated by other teams.

Relatively low number of lines of code for test suites

The number of lines of tests is only one fifth of the number of lines of code. This does not necessarily demonstrate under-testing, but is a strong indication. Further evidence would be provided by test coverage reports, which are currently lacking.

```
cloc */test test
      87 text files.
      79 unique files.
      10 files ignored.

1 error:
Unable to read:  test

github.com/AlDanial/cloc v 1.74  T=0.16 s (488.2 files/s, 62836.8 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           co
-------------------------------------------------------------------------------
Haskell                         75           1316           1044           74
Bourne Shell                     2             31              6
-------------------------------------------------------------------------------
SUM:                            77           1347           1050           75
-------------------------------------------------------------------------------
```

Versus

```
$ cloc */src src
     389 text files.
     389 unique files.
       0 files ignored.

1 error:
Unable to read:  src

github.com/AlDanial/cloc v 1.74  T=0.63 s (620.9 files/s, 87357.2 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           co
-------------------------------------------------------------------------------
Haskell                        388           7545           7076          400
YAML                             1             17             34
-------------------------------------------------------------------------------
SUM:                           389           7562           7110          400
-------------------------------------------------------------------------------
```

First occurrence: January 2018

Status: In progress

IOHK Response

> More resources have been allocated to increase test coverage and quality. We are constantly increasing and improving our test coverage as the complexity of our platform grows. Testing and quality have been, are, and always will be high priorities.

● The project uses the `error` function instead of correctly dealing with failure cases

There are more than one hundred occurrences of `error` or `undefined` in the project. These functions are often used to handle failure cases without explicit type-level indication of the potential error. We recommend against using these functions because they can lead to unpredictable failure cases as a result of how they interact with laziness.

Instead, it is highly recommended to structure the data types to prevent impossible code paths from being reached. For cases which may fail at runtime, we recommend either using a datatype (such as `Either`) to represent the potential failure, or using a proper runtime exception type, with the possibility of failure expressed by the usage of `IO` or `MonadIO`.

Note that after brief review, we believe the policy document provided in IOHK's response represents a sensible and useful approach to this topic.

First occurrence: February 2018

Status: In progress

IOHK Response

> We are in the process of addressing this issue, but still have some legacy occurrences in the project. The policy that reviews this issue can be found here: https://github.com/input-output-hk/cardano-sl/blob/develop/docs/exceptions.md
>
> Our new policy describes all exception handling, including when it is and is not appropriate to use `error`. Work to bring the code into compliance with the new policy is ongoing. It is worth noting that the policy does allow for the use of error in some cases.

● Using the non-best practice `Given` for configuration instead of `ReaderT`

The `Data.Reflection.Given` type is used pervasively throughout the codebase. It is not considered a best practice in Haskell development. By contrast, usage of `ReaderT` for this purpose is well-established in production codebases.

There may be issues with the `Given` approach that have not been discovered due to the fact that this is not a frequently used approach. Indeed, the documentation for `give` already points toward a potential problem:

```
give :: forall a r. a -> (Given a => r) -> r

Reify a value into an instance to be recovered with given.

You should only give a single value for each type.
If multiple instances are in scope, then the behavior
is implementation defined.
```

This type of problem could not exist when using the `ReaderT` approach.

First occurrence: January 2018

Status: Acknowledged

IOHK Response

> The developers acknowledge that this practice is not ideal. We are in the process of changing this to explicit value passing or via Reader monad where appropriate. We have already done this for the network subsystem for example. This is a minor issue.

Over-use of monad transformers

The project uses a large number of different monad transformers. Given that their instances are not free and difficult to test, we recommend against this approach. Note that the project has already suffered from bugs due to incorrectly implemented monad transformers in the past. We are concerned that similar problems will show up in the future.

First occurrence: January 2018

Status: Low priority

IOHK Response

> Developers are in the process of refactoring the monad stack. This task will not completed within the current month. This is minor and does not have any impact on performance, security or reliability of the platform.

Some partial functions (e.g. `decodeUtf8`) from `text` are not hidden

As covered in its own flag, the custom prelude `Universum` does a very good job of hiding partial functions from `base`. However, it does not hide some other partial or otherwise unsafe functions

- From `Universum.String.Conversion`, a partial function `decodeUtf8`
- From `Universum.Lifted.File`, a function `getContents` using lazy I/O
- From `Universum.Lifted.File`, a function `readFile` using lazy I/O and implicit character encoding handling

The `decodeUtf8` function was one of the causes of a vulnerability discovered in the `cborg` library. Implicit character encoding handling is a common cause of platform-specific bugs which are difficult to detect with standard testing (see blog post Beware of readFile). We are concerned all of these could be the cause of currently undetected vulnurabilities.

First occurrence: January 2018

Status: Minor issue

IOHK Response

> For decodeUTF8, IOHK will consider adding it to a hlint rule so that it is flagged up and can be allowed on a case-by-case basis (by hlint pradmas). For the other functions, we believe that there are legitimate uses of these functions, and we plan to deal with any inappropriate uses by means of a code review. This is a minor issue. This is a minor issue.

Use of partial functions in the project

The project uses partial functions such as:

- `fromJust`
- `maximum`
- `minimum`
- `decodeUtf8`

- `pred`
- `succ`
- `toEnum`

Using these functions means allowing for the possibility of getting unexpected errors with unhelpful error messages like the following:

```
*** Exception: Prelude.minimum: empty list
```

Furthermore, partial functions decrease the strength of static code guarantees provided by the Haskell type system. We can no longer say for certain, for example, that a value of type `Int` actually contains an `Int`, or may also contain an exception.

Note that `decodeUtf8` in `cborg` has already caused a red flag.

Note that we have reviewed some usages of these functions in the codebase, and these usages did not look specifically unsafe. That is why this is only a yellow flag.

First occurrence: February 2018

Status: Minor issue

IOHK Response

> We acknowledge that these usages are not ideal, but the issue is rather minor and we will address it on a case-by-case basis. We do not think that it is legitimate to completely ban the use of partial functions.

● Custom prelude

The custom `Universum` seems like very good replacement for the standard prelude. These custom defaults can prevent many mistakes caused by bad defaults. It is certainly a worthwhile investment.

In addition to the above comment from the last report, there are some particularly good aspects that we would like to highlight:

- Compile-time warnings for debugging functions in `Universum.Debug`.
- The bug function in `Universum.Exception` is much better-named than `error` and guaranteed to contain a callstack.
- The `Universum.Container.Class` hierarchy forbids usages of the `length` function over tuples and `Maybe` values.
- Re-exports use a whitelist strategy instead of a blacklist strategy.

First occurrence: December 2017

● Partial functions from base are not exposed

Common partial functions like head, `tail`, `init` and `last` from the standard prelude are not exposed the Universum prelude.

This prevents standard crashes with obscure errors like the following:

```
*** Exception: Prelude.head: empty list
```

First occurrence: January 2018

● Many standard mistakes are eliminated by Universum

As one concrete example, Universum uses strict versions of `sum` and `product`. This eliminates a potential space leak with the default `sum` and `product` functions.

First occurrence: February 2018

Detailed flags: Code management

	Public	Private	Total
Old flags	4	0	4
New flags	1	0	1
Total flags	5	0	5
Newly resolved flags	0	0	0
Red flags	1	0	1
Yellow flags	2	0	2
Green flags	2	0	2

Many bugs lack regression testing in resolution

Regression tests are important to make sure that problems that have been detected and solved will not recur in the future undetected. This is particularly important for potential security vulnerabilities.

From our analysis, the following pull requests are some examples of fixes that do not introduce corresponding regression tests into the test suite:

- PR 2460
- PR 2172
- PR 2113
- PR 2066
- PR 1885

While the particular instances listed above deserve attention, the broader point of this note is to comment on the engineering process. It is our recommendation to institute engineering processes which encourage best practices, including regular introduction of regression tests to the project's test suite. One example of such a rule may be:

> All bug fixes must meet these requirements:
>
> - be submitted as a pull request
> - refer to the issue being addressed
> - receive sign-off from the original issue reporter (if part of the team)
> - either introduce a new automated test which triggers the bug and is fixed by the pull request, include written instructions for a manual test to trigger the bug, or include a justification for why such methods would be inappropriate in this case (with the first choice, an automated test, being most preferred)

First occurrence: February 2018

Status: In progress

IOHK Response

> As part of one of our QA improvement initiatives for SDLC and STLC processes we are building out and improving our documented regression library set. We are changing the acceptance criteria for the bug fixing process to include regression testing we want to ensure that we introduce this improvement without disruption.

● Daedalus does not pin a version of the `cardano-sl` codebase

The code in the `daedalus` module interoperates with the code in the `cardano-sl` module, which means that if the commits are not pinned, these two could drift apart in incompatible ways.

Initially, our report recommended making `daedalus` a Git submodule of the `cardano-sl` codebase. Following response from IOHK, we now understand that this relationship should work the other way around.

First occurrence: December 2017

Status: In progress

IOHK Response

> This item should be moved to the Recommendations section.
>
> Daedalus should not be a submodule. It pulls the cardano-sl binaries and as a result we will pin the cardano-sl commit revision in Daedalus. This issue is being tracked with DEVOPS-673: will pin down cardano-sl to a revision in Daedalus (and not vice versa).

● The `master` branch is not the default branch to clone on Github

The `develop` branch is set to be the default branch to clone on Github. This means that the project encourages users of the source code to use a bleeding edge version instead of the latest *stable* version.

First occurrence: January 2018

Status: Will review later

IOHK Response

> The development branch is more suitable for contributors, that is why it is set as the default branch. There is a tradeoff here between defaulting to the develop or master branch. The develop branch is more suitable for external contributors while the master branch is more suitable for external users simply wanting to build the current version. We will keep this decision under review, based on user feedback.

● The project uses git-flow

This git branching model is considered a best practice in open-source development.

First occurrence: December 2017

● There exists a document that describes the git workflow

There exists a public document that outlines the release strategy for cardano.

First occurrence: January 2018

Detailed flags: Continuous integration

	Public	Private	Total
Old flags	8	0	8
New flags	0	0	0
Total flags	8	0	8
Newly resolved flags	0	0	0
Red flags	1	0	1
Yellow flags	5	0	5
Green flags	2	0	2

● Opt-in nature of CI may lead to undiscovered test failures

In the current Continuous Integration (CI) setup, each new package added to the project must be explicitly added to the CI scripts. Such a situation can lead to a false sense of security due to lack of test suite failures. In fact, lack of test suite failures may only indicate that the test suite was not added to the list of tests.

Instead, we recommend automatically generating the list of packages to test using such techniques as the `stack ide-targets` command. In addition, a script like the CICheck script used in writing this report can be used to verify the test results post-hoc. Finally, keeping metrics on the test suite runs (such as number of tests run, or code coverage) can help indicate if tests are no longer being run.

First occurrence: January 2018

Status: In progress

IOHK Response

> As you replied to our request for clarifications: “Currently, the CI scripts require an “opt in” style approach for choosing the packages on which to run their test suites. This can lead to a situation where CI passes, not because all of the tests pass, but because the tests were never run. Instead, it would be ideal if all packages are automatically added to the CI's test runs. A step down from that would be some verification that all of the test suites have been run, even if they remain opt-in. Clarifications have been made to the wording of this flag for the upcoming report to make it clearer.“ Feedback from our DevOps team is the following: “IOHK DevOps require clarification on which CI scripts they are referring to as appveyor/buildkite have different logic. In appveyor, we call `stack install cardano-sl cardano-sl-tools cardano-sl-wallet cardano-sl-wallet-new --test`, whereas, in buildkite we enable all tests for local packages in stack.yaml. On darwin, we disable tests for tools and cardano-sl due to the load-command size, but we can check if that works, but that is ‘opt-out’.” It seems that additional feedback is required. It's a valid concern for appveyor (windows). We will convert testing within this CI system from 'opt-in' to 'opt-out'. It is a valid concern for appveyor (Windows). This is being tracked by DEVOPS-725.

● Haddock API documentation fails to build

This means that developers cannot browse the Haskell documentation locally. It may also be indicative of a larger issue: docs are not being used for development. In particular, this could

mean that packages are used by looking up the API via the source code instead of the documentation.

Evidence:

```
--  While building package rocksdb-haskell-ng-0.0.0 using:
  /home/fpc/.stack/setup-exe-cache/x86_64-linux-nix-tinfo6/Cabal-simple_mPHDZ
J_1.24.2.0_ghc-8.0.2 --builddir=.stack-work/dist/x86_64-linux-nix-tinfo6/Caba
1.24.2.0 haddock --html --html-location=../$pkg-$version/ --haddock-option=--
perlinked-source
Process exited with code: ExitFailure 1
Logs have been written to: /home/fpc/cardano-sl/.stack-work/logs/rocksdb-hask
l-ng-0.0.0.log

Preprocessing library rocksdb-haskell-ng-0.0.0...
Running Haddock for rocksdb-haskell-ng-0.0.0...
Preprocessing library rocksdb-haskell-ng-0.0.0...

src/Database/RocksDB.hs:11:5: error:
    parse error on input ‘-- * Basic Functions’
```

First occurrence: December 2017

Status: Reported Complete

IOHK Response

> The following PR was merged on 01.02.2018: https://github.com/input-output-hk/cardano-sl/pull/2410

The CI for master isn't all-green

If we look at the history of the builds for the master branch on travis, we can see that not all of the CI builds have passed. This issue may not be apply to the buildkite logs. At present, however, those are not available to us, so we cannot verify this.

Given that `origin/master` is supposed to represent a production-ready state of the system, this should be remedied.

First occurrence: January 2018

Status: Reported Complete

IOHK Response

> Travis CI (the one that was used previously and encountered these problems) has been retired. All the other CI systems (such as BuildKite, appVeyor, and Hydra) do not encounter such problems. Please point out the specific case where this issue is still valid.

Buildkite logs are not public

In the transition from travis to buildkite, CI logs became a private artifact. Ideally the community should have access to the CI logs so they can independently verify that the CI tests what they expect it to test.

First occurrence: January 2018

Status: Blocked by vendor

IOHK Response

> This issue is being addressed by the developers. We are waiting for buildkite development team to implement this feature. Please refer to: https://github.com/buildkite/feedback/issues/137#issuecomment-360336774
>
> The reason this is the case is that buildkite logs can contain secrets. We are working with the buildkite team as they will soon release 3.0 to address these issues.

● Test coverage is not run in CI

It is essential to have an overview of which parts of the code are not well tested.

A test coverage report provides automatic feedback about whether the testing of any part of the code has been forgotten.

First occurrence: December 2017

Status: In backlog

IOHK Response

> The developers consider the HPC tool (link) as a solution to this matter. This issue is in the backlog.

● Haddocks are not built in CI

Haddock is the standard API documentation system used for Haskell code. Such API level documentation is regularly used by software developers to understand a code base and assist in contributing to it.

In the generic builder for Haskell packages in Nix, doHaddock is `false` by default: https://github.com/NixOS/nixpkgs/blob/82f626702314928c9c8f4ea309430e3aa0680d57/pkgs/development/haskell-modules/generic-builder.nix#L45 In the project repository, doHaddock is never set to true:

```
$ ag doHaddock | grep true
$ echo $?
1
```

There is a part in the `scripts/ci/ci.sh` script that suggests that Haddocks should be generated on the master branch:

```
if [[ ("$OS_NAME" == "linux") && ("$BUILDKITE_BRANCH" == "master") ]];
  then with_haddock=true
  else with_haddock=false
fi
```

However, it does not appear that this variable is used anywhere:

```
grep -r "with_haddock" .
./scripts/ci/ci.sh:  then with_haddock=true
./scripts/ci/ci.sh:  else with_haddock=false
```

It is also recommended to build Haddocks on non-master branches as well, since Haddocks should always be buildable for use during development.

Building Haddocks on CI ensures that any contribution preserves the invariant that Haddocks can be built. This should be turned back on as soon as possible.

Note that the Haddocks have already been broken and fixed twice. Without adding Haddocks to the CI run, there is no guarantee that the Haddock build will not be broken again.

- PR 2410
- PR 2072

First occurrence: December 2017

Status: In backlog

IOHK Response

> This issue is in the backlog.

● The project uses Nix on CI

This makes CI both reproducible and composable.

First occurrence: January 2018

● The project uses CI

CI ensures a level of quality for code added to the project, and automatically detects certain classes of regressions.

First occurrence: January 2018

Detailed flags: Dependencies

	Public	Private	Total
Old flags	5	3	8
New flags	1	0	1
Total flags	6	3	9
Newly resolved flags	0	0	0
Red flags	0	2	2
Yellow flags	6	1	7

Duplicated solutions

In a number of cases, the project depends on multiple dependencies which solve the same or similar problems. These may not be redundant, but this should be investigated.

- Preludes: `universum`, `base-prelude` and `foundation`.
- Effectful Streaming: `pipes` and `conduit`.
- Parsing: `split`, `polyparse`, `parsec`, `parser-combinators`, `megaparsec`, `attoparsec`
- Interpolation: `wl-pprint-text`, `text-format`, `pretty`, `neat-interpolation`, `formatting`, `fmt`, `ansi-wl-pprint`
- Lenses: `lens and microlens`
- Time: `time`, `time-units`, `time-compat`, `old-time`, `old-locale` `unix-time`, `time-locale-compat` and `hourglass`.
- Binary serialisation: `cborg`, `cereal` and `binary`.
- Paths: `system-filepath`, `filepath`
- File operations: `directory` and `easy-file`
- String conversions: `string-conv` and `string-conversions`
- Exceptions: `safe-exceptions`, `exceptions`, `extensible-exceptions`
- Randomness: `mwc-random`, `random`
- Monad morphisms: `natural-transformation`, `mmorph`
- Logging: `log-warper`, `fast-logger`

First occurrence: December 2017

Status: In progress

IOHK Response

> Solutions to this issue will be continuously rolled-out over the next few months.

No evidence of vetting of dependencies

- There should be a (public) process for vetting dependencies for safety and security problems.
- There should be a (public) log of evidence of this vetting with the specific versions that were used.
- There should be a (public) process to address the problems that arise in this vetting process

A starting point for the vetting process includes checking:

- The maturity of the library
- How much the library is being used
- How many open issues there are with bugs
- Whether, and how well, the library is maintained

- What kind of reputation the library has

First occurrence: December 2017

Status: In progress

IOHK Response

> Solutions to this issue will be continuously rolled-out over the next few months.

The project uses relatively untested log-warper

`log-warper` is a young project with, in our opinion, insufficient real world testing to justify its usage, when other alternatives exist. Basic investigation into the library indicates that it may have serious flaws. As an example, it has the following comment:

```
-- TODO: correct exceptions handling here is too smart for me
```

First occurrence: January 2018

Status: In progress

IOHK Response

> Plans have been made to move to structured logging, which should solve this problem.
>
> We plan to replace this with logging based on the katip package.

Blocks are inefficiently stored in individual files

Currently, each block is stored in an individual file. However, as documented at https://github.com/input-output-hk/cardano-sl/issues/2224, this has disadvantages. There is a proposal to fix this issue at: https://github.com/input-output-hk/cardano-sl/blob/535c36cf9496958e96aabf57bb875012060b3b34/docs/proposals/block-storage.md. From the point of view of code simplicity and reliability, however, it would be better to use preexisting solutions designed to solve these problems.

One example would be SQLite, which explicitly documents its abilities to replace multiple small files on a filesystem with a single database: https://www.sqlite.org/fasterthanfs.html. There may be mitigating factors preventing the usage of such a more commonly used storage technology, but there is no evidence of such a conclusion having been reached.

Barring such evidence, our conclusion is that the current storage methodology, and plans for mitigating current risks, introduces undue risk to the project in terms of code complexity, potential data loss, and logic errors.

First occurrence: February 2018

Status: Acknowledged

IOHK Response

> We acknowledge that the use of many small files is not a good long term design choice for several reasons. It was an expedient choice during rapid development. We are re-analysing the requirements of the whole storage subsystem, both block storage and associated indexes, and will then choose a new design. This process, including migration, must be done properly and we anticipate that it will take considerable time.

Usage of non-transactional data storage layer

There is no documentation as to why Rocksdb was chosen. From our review of the technical requirements, it appears that Rocksdb is not a good technical choice for this project:

- The standard ACID guarantees provided by other databases are both desirable for this project, and absent in Rocksdb.
- The project is relatively young.
- There exists a *de facto* standard technology for the case of embedded database: SQLite. When such a de facto standard exists, we would expect to see a rationale for using an alternative.

The RocksDB FAQ clearly states:

```
Q: Can I write to RocksDB using multiple processes?

A: No. However, it can be opened in read-only mode from multiple processes.
```

```
Q: Is it safe to close RocksDB while another thread is issuing read, write or
manual compaction requests?

A: No. The users of RocksDB need to make sure all functions have finished
before they close RocksDB.
```

The above points mean that that great care has to be taken when using RocksDB in a multithreaded context. Any crash could cause memory corruption if it happens while rocksdb is writing.

First occurrence: December 2017

Status: Will review later

IOHK Response

> In the context of cryptocurrencies the use of RockDB is not non-standard or unusual. Bitcoin itself uses LevelDB, and RocksDB is simply a fork of LevelDB maintained by Facebook. RocksDB does not support full composable transactions but it does support atomic batched updates, and the code makes extensive use of that. We are also careful not to update the DB from more than one thread concurrently. That said, the choice of RocksDB is certainly debatable. We are currently reassessing the requirements and design for the whole storage subsystem, including any database. As an outcome of this we may choose not to use RocksDB.

The project uses a non-standard variant of JSON

Canonical JSON, as implemented by the `hackage-security` library, is not a subset of JSON. The module itself acknowledges this:

```
NB: Canonical JSON's string escaping rules deviate from RFC 7159 JSON which
requires

"All Unicode characters may be placed within the quotation marks, except for
the characters that must be escaped: quotation mark, reverse solidus, and the
control characters (U+0000 through U+001F)."
```

```
Whereas the current specification of Canonical JSON explicitly requires to
violate this by only escaping the quotation mark and the reverse solidus. Thi
however, contradicts Canonical JSON's statement that "Canonical JSON is
parsable with any full JSON parser"

Consequently, Canonical JSON is not a proper subset of RFC 7159.
```

Interoperation with real JSON will cause problems.

Canonical JSON uses `Int54` because “Javascript can only safely represent numbers between `-(2^53 - 1)` and `(2^53 - 1)`”, but canonical JSON already cannot interoperate with Javascript because of the first point.

`Int54` is implemented without any bounds checking, so it is effectively just an `Int64`. There is a comment about this:

```
TODO: Although we introduce the type here, we don't actually do any bounds
checking and just inherit all type class instance from Int64. We should
probably define fromInteger to do bounds checking, give different instances f
type classes such as Bounded and FiniteBits, etc.
```

The parser for `Int54` is not sound with respect to the `Int54` bounds either, it just parses fifteen digits instead of integers from the correct range. There is a comment about this:

```
-- TODO: Currently this allows for a maximum of 15 digits (i.e. a maximum val
-- of @999,999,999,999,999@) as a crude approximation of the 'Int54' range.
```

It cannot handle unicode code-points U+00ff There is a comment about this:

```
TODO: Known bugs/limitations:

Decoding/encoding Unicode code-points beyond U+00ff is currently broken
```

It uses `[Char]` for parsing. This may potentially result in slow parsing, which will at least impact the performance of the cardano-sl codebase, and at worse provide a means of further DoS attacks through carefully crafted input.

There are roundtrip tests in the test suite, but no tests for adversarial inputs.

First occurrence: January 2018

Status: Will not fix

IOHK Response

> This would be a problem if full interoperability with normal JSON were needed, or if the canonical JSON decoder could be used with a very large amount of input data. However that is not the case in the way the canonical JSON library is being used in Cardano SL. It is used only to decode the genesis block. The genesis block is fixed and shipped with the software. The use of canonical JSON is to allow for a transparent and auditable genesis block format, using an open standard, but in a way that has stable hashes (for example, insensitive to whitespace).

Detailed flags: Documentation

	Public	Private	Total
Old flags	7	0	7
New flags	0	0	0
Total flags	7	0	7
Newly resolved flags	0	0	0
Yellow flags	5	0	5
Green flags	2	0	2

🟡 The building/contribution guide does not work on all systems

I followed the `docs/how-to/build-cardano-sl-and-daedalus-from-source-code.md` guide and it works wonderfully now.

Now both the 'Nix' workflow and the 'Stack with Nix' workflow work well. The 'Daedalus Wallet' workflow still does not work flawlessly.

When one follows the guide to the letter, they get the following error.

```
$ npm install
bash: npm: command not found
```

Given that the project uses `nix`, `npm` should certainly be configured in nix and not assumed from the global system. This will also help with potential reproducibility problems.

When one installs `npm` globally to work around the problem in the previous section, the 'Building Daedalus' section still does not work as intended:

```
$  npm install
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libssl.s
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libssl.s
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libssl.s
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libcrypt
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libcrypt

> daedalus@0.8.0 preinstall /home/fpc/daedalus
> which electron || npm install electron@1.7.9

node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libssl.s
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libssl.s
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libssl.s
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libcrypt
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libcrypt
/home/fpc/daedalus/node_modules/.bin/electron
npm WARN deprecated babel-preset-es2015@6.24.0: 🔠  Thanks for using Babel: w
npm WARN deprecated babel-preset-es2015@6.24.1: 🔠  Thanks for using Babel: w
npm WARN deprecated babel-preset-latest@6.16.0: We're super 😸 excited that y
npm WARN deprecated babel-preset-es2016@6.24.1: 🔠  Thanks for using Babel: w
npm WARN deprecated babel-preset-es2017@6.24.1: 🔠  Thanks for using Babel: w
npm WARN deprecated fs-promise@2.0.3: Use mz or fs-extra^3.0 with Promise Sup
```

```
npm WARN deprecated tar.gz@1.0.7: ⚠ WARNING ⚠ tar.gz module has been deprecate
npm WARN deprecated minimatch@0.3.0: Please update to minimatch 3.0.2 or high

> scrypt@6.0.3 preinstall /home/fpc/daedalus/node_modules/scrypt
> node node-scrypt-preinstall.js

node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libssl.s
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libssl.s
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libssl.s
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libcrypt
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libcrypt

> scrypt@6.0.3 install /home/fpc/daedalus/node_modules/scrypt
> node-gyp rebuild

node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libssl.s
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libssl.s
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libssl.s
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libcrypt
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libcrypt
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libssl.s
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libssl.s
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libssl.s
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libcrypt
node: /nix/store/cr49q6vvyhfs48dg15ys3mghfzgc80i7-openssl-1.0.2l/lib/libcrypt
gyp: Call to 'node -e "require('nan')"' returned exit status 0 while in bindi
gyp ERR! configure error
gyp ERR! stack Error: `gyp` failed with exit code: 1
gyp ERR! stack     at ChildProcess.onCpExit (/usr/lib/node_modules/npm/node_m
gyp ERR! stack     at ChildProcess.emit (events.js:160:13)
gyp ERR! stack     at Process.ChildProcess._handle.onexit (internal/child_pro
gyp ERR! System Linux 4.14.15-1-ARCH
gyp ERR! command "/usr/bin/node" "/usr/lib/node_modules/npm/node_modules/node
gyp ERR! cwd /home/fpc/daedalus/node_modules/scrypt
gyp ERR! node -v v9.4.0
gyp ERR! node-gyp -v v3.6.2
gyp ERR! not ok
npm WARN daedalus@0.8.0 No repository field.
npm WARN daedalus@0.8.0 No license field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.1.3 (node_modules/
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fseven

npm ERR! code ELIFECYCLE
npm ERR! errno 1
npm ERR! scrypt@6.0.3 install: `node-gyp rebuild`
npm ERR! Exit status 1
npm ERR!
npm ERR! Failed at the scrypt@6.0.3 install script.
npm ERR! This is probably not a problem with npm. There is likely additional

npm ERR! A complete log of this run can be found in:
npm ERR!     /home/fpc/.npm/_logs/2018-02-02T05_22_14_757Z-debug.log
```

This part should also be run on CI so that this can never occur for contributors.

We have not been able to look into the rest of the guide as a result of the problems above.

First occurrence: December 2017

Status: In progress

IOHK Response

> The only issue reported within the scope of this item is 'Daedalus Wallet' build guide. We are currently testing this packaging and documentation will be updated once it is done.

Some documents have no owner

To prevent documents from becoming—and subsequently remaining—out-of-date, it is important that each document is clearly marked with an owner. This person is then responsible for keeping the document up-to-date, just like a code owner is responsible for the code they own.

As concrete examples of these findings:

- We are currently treating the Project Charter as the formal requirements for the Shelley release. The copy of the document provided to us is version 1.0, by Bryan Tierney. To our knowledge, Bryan is no longer a member of the IOHK team.
- The Git repositories contain many Markdown documentation files. For example, cardano-sl has a `/docs` directory, and the cardanodocs.com repo contains many more. None of these documents explicitly list an owner. Note that, in particular, the cardanodocs.com repo already uses Markdown's built in metadata support on Github, and could therefore include an `owner` metadata field as well.
- The release-process.pdf document provided indicates a “last edited” field, but no official owner for the document.
- The “IOHK Requirements Management” document contains no owner field.

First occurrence: January 2018

Status: In progress

IOHK Response

> This does need to be addressed, as does the lack of owner on the .md files. New templates are being rolled out across the company which have the owner listed on the front page, plus a document revision table has been added to all document templates to provide visibility on all updates.

Links to internal documents in the public documentation

Many documents in the `cardano-sl` repository mention documents that are not open-source. Examples include documents on the issue tracker, like `CSL-1025` and even explicit links like https://iohk.myjetbrains.com/youtrack/issue/CSL-1025 in `docs/proposals/block-storage.md`.

First occurrence: January 2018

Status: In progress

IOHK Response

> References will be removed or moved to a public domain. This will be done as references are encountered.

There are runtime errors in the demo

When one follows the instructions in `scripts/README.md` to run the demonstrations, they may be greeted by the following error:

```
$ ./scripts/launch/demo.sh
+ '[' -n '' ']'
+ echo 'You must run this script from the tmux session!'
You must run this script from the tmux session!
+ exit 1
```

As a result, the demo cannot be run without meeting undocumented requirements. The demo uses `tmux`, which is not a standard tool and not managed by nix. Installing `tmux` globally still does not allow everyone to run the demo, because `tmux` will use the global shell and its initialisation scripts.

One potential resolution would be to use `docker compose` for running the demo. It is also important that the demo is run in CI to make sure it continues to work in the future.

There are public issues about this.

- https://github.com/input-output-hk/cardano-sl/issues/2451
- https://github.com/input-output-hk/cardano-sl/issues/2217

First occurrence: December 2017

Status: Acknowledged

IOHK Response

> This will be investigated.

There is no Readme for most packages, and no overview of packages

In order to make onboarding smooth, there should be a clear overview of all packages in the repository. This overview should outline the owners of each package and the responsibilities of the package.

To find the directories that need a `README.md` (but may already have one), we ran the following command:

```
find . -maxdepth 1 -not -path '*/\.*' -type d
.
./lrc
./wallet
./crypto
./wallet-new
./docs
./node
./scripts
./pkgs
./networking
./secrets
./tools
./util
./ssc
```

```
./node-db
./delegation
./db
./lib
./run
./client
./explorer
./wallet-db
./block
./generator
./txp
./auxx
./update
./logs
./binary
./infra
./core
```

To find the directories that *have* a `README.md`, we ran a similar command:

```
find . -maxdepth 2 -type f -iname 'README.md'
./wallet-new/README.md
./docs/README.md
./node/README.md
./scripts/README.md
./networking/README.md
./lib/README.md
./explorer/README.md
```

First occurrence: December 2017

Status: Acknowledged

IOHK Response

> This issue has been resolved by our Quality and Documentation teams in response to the first audit. READMEs have been added for the majority of components.

● There is a public roadmap document

There is a roadmap document (https://cardanoroadmap.com/), and it is publicly available. This provides significant transparency to the community.

First occurrence: January 2018

● Each project management document has a 'purpose' section

All project management documents that we have received have a clear 'purpose' section.

First occurrence: January 2018

Detailed flags: Quality

	Public	Private	Total
Old flags	1	0	1
New flags	4	2	6
Total flags	5	2	7
Newly resolved flags	0	0	0
Red flags	0	2	2
Yellow flags	2	0	2
Green flags	3	0	3

Insufficient variety in testing approaches

From our review, the current set of tests mostly focus on “success test” cases. This includes even security-sensitive parts of the code, where regressions have been discovered previously. We strongly recommend expanding testing in significant ways. This would include, at the least, failure tests and security tests. Scripted, manual testing from quality engineers would also be highly valuable.

Aaron Contorer, CEO of FP Complete, has written a blog post on variety in testing: Devops Best Practices: Multifaceted Testing.

First occurrence: February 2018

Status: Acknowledged

IOHK Response

> We are starting to perform this coverage with the negative tests for the serialization library (cborg). We acknowledge that there are insufficient test properties that cover negative cases. In particular fuzz testing would be a useful addition. We are constantly increasing the coverage of negative cases as part of development of tests for new features and improving the quality of tests of existing features.

No evidence received of security-focused testing

The mathematical models, network architecture, and code design for the project are focused on providing high levels of security. There are specific attack vectors called out, and mitigations have been designed and implemented. However, to our knowledge, there is no set of tests currently in place to exercise the efficacy of these designs.

We would expect to see a variety of tests design to find flaws in the system, including, but not limited to:

- Simulated DoS attacks
- “Chaos Monkey” style cloud provider failures
- Simulated netsplits
- Malformed input submission

First occurrence: February 2018

Status: Acknowledged

IOHK Response

> The issue raised here is partly about security and partly about resilience/robustness e.g. DoS attacks and comms failures.

> For the security aspect, negative input and fuzz testing is useful but not sufficient. For security our focus is on formalising specifications and code audit. We have a dedicated internal security audit team who have reviewed all major security critical aspects of the code prior to the mainnet release and ongoing thereafter.
>
> For robustness testing, comms failures etc, our plan is to address this in simulation as part of the Shelly networking work (ie tests with a simulated network layer where it is quick and easy to simulate communication failures).
>
> For one aspect of DoS attacks, we have already have system level benchmarks that observe the system behaviour when the relays are overloaded with transaction requests.

Static verification of code correctness

The project has selected software methodologies which encourage static verification. This is primarily achieved by usage of the Haskell programming language. While not all classes of bugs can be addressed via a strong type system, many can, when following proper coding discipline. Haskell is particularly well-suited for cardano-sl because of the high safety and security requirements.

First occurrence: January 2018

Memory safety by default

The software methodology followed uses, by default, memory safe paradigms. Such approaches to software development avoid some of the most severe types of vulnerabilities, such as buffer overruns. The majority of the codebase lives within the “memory safe” subset of Haskell, providing almost complete protection against entire classes of bugs.

NOTE Haskell does allow memory-unsafe access, via its Foreign Function Interface (FFI). There is also the potential for bugs in underlying technologies, such as the kernel, the C library, or the runtime system. Therefore, the project is not *immune* to such bugs, but the surface area for such mistakes is greatly reduced.

First occurrence: February 2018

Well structured Byron Updates test plan

While the Byron Updates release is *not* in scope for this audit, we have been provided with the test plan for this release as an example of potential future Shelley-related test plans. We have reviewed this plan, from the standpoint of a general test plan. Our finding is that this is a well designed plan, demonstrating professional quality assurance practices.

Note that, at the same time, the report indicates insufficient test coverage until now, and the lack of a test plan for the Shelley release. These points, however, are covered in other flags.

First occurrence: February 2018

Appendix

The project uses the `error` function instead of correctly dealing with failure cases

`hlint-error.yaml`

```
- extensions:
  - name: [TypeApplications]
- functions:
  - {name: error, within: []}
  - {name: undefined, within: []}
```

`run_hlint_error.sh`

```
#!/bin/bash

cd ~/cardano-sl

hlint . --hint=../cardano-reports/hlint-error.yaml --only "Avoid restricted f
```

Use of partial functions in the project

`hlint.yaml`

```
- extensions:
  - name: [TypeApplications]
- functions:
  - {name: unsafeInterleaveIO, within: []}
  - {name: unsafePerformIO, within: []}
  - {name: unsafeCoerce, within: []}

  - {name: div, within: []}
  - {name: quot, within: []}
  - {name: rem, within: []}
  - {name: quotRem, within: []}
  - {name: mod, within: []}

  - {name: succ, within: []}
  - {name: pred, within: []}
  - {name: toEnum, within: []}
  - {name: '^', within: []}

  - {name: hPutStr, within: []}

  - {name: readFile, within: []}
  - {name: writeFile, within: []}

  - {name: decodeUtf8, within: []}
```

```
  - {name: fromJust, within: []}
  - {name: fromLeft, within: []}
  - {name: fromRight, within: []}

  - {name: minimum, within: []}
  - {name: maximum, within: []}

  - {name: read, within: []}

  - {name: '!', within: []}
```

run_hlint.sh

```
#!/bin/bash

cd ~/cardano-sl

hlint . --hint=../cardano-reports/hlint.yaml --only "Avoid restricted functio
```

Opt-in nature of CI may lead to undiscovered test failures

CICheck.hs

```
module CICheck
    ( ciCheck
    ) where

import Data.List
import Data.Maybe
import System.Process

ciCheck :: IO ()
ciCheck = do
    (_, _, err) <- readProcessWithExitCode "stack" ["ide", "targets"] ""
    let targets = mapMaybe parseTarget $ lines err
    log <- getContents
    putStrLn $
        unlines $
        flip map targets $ \t ->
            let b =
                    ((`isInfixOf` log) . unwords) $
                    case t of
                        Lib p -> ["library", p]
                        Test p t -> ["test suite", "'" ++ t ++ "'", "for"]
                        Bench p t -> ["benchmark", "'" ++ t ++ "'", "for"]
                        Exe p t -> ["executable", "'" ++ t ++ "'", "for"]
                s =
                    if b
                        then "GOOD"
                        else "BAD "
```

```
                u = intercalate ":"
            in unwords
                   [ s
                   , case t of
                         Lib p -> u [p, "lib"]
                         Test p t -> u [p, "test", t]
                         Bench p t -> u [p, "bench", t]
                         Exe p t -> u [p, "exe", t]
                   ]

type Pkg = String

type TestSuite = String

type Benchmark = String

type Executable = String

data Target
    = Lib Pkg
    | Test Pkg
           TestSuite
    | Bench Pkg
            Benchmark
    | Exe Pkg
          Executable
    deriving (Show, Eq)

parseTarget :: String -> Maybe Target
parseTarget s = do
    let (before, _after) = span (/= ':') s
    after <-
        case _after of
            [] -> Nothing
            (':':a) -> Just a
    let (typ, _rest) = span (/= ':') after
    let rest =
            case _rest of
                [] -> Nothing
                (':':r) -> Just r
    case typ of
        "lib" -> Just $ Lib before
        "test" -> Test before <$> rest
        "bench" -> Bench before <$> rest
        "exe" -> Exe before <$> rest
        _ -> Nothing
```

Statistics

	Public	Private	Total
Old flags	32	5	37
New flags	9	2	11
Total flags	41	7	48
Newly resolved flags	0	0	0
Red flags	2	5	7
Yellow flags	27	2	29
Green flags	12	0	12