Technical Review of Djed

High Assurance Software Group
November 29, 2022

Contents

Chapter 1

Executive Summary and Scope

This report is presented without warranty or guaranty of any type. This report lists the most salient concerns that have so far become apparent to Tweag after a partial inspection of the engineering work. Corrections, such as the cancellation of incorrectly reported issues, may arise. Therefore Tweag advises against making any business decision or other decision based on this report.

Tweag does not recommend for or against the use of any work or supplier referenced in this report. This report focuses on the technical implementation provided by the project's contractors and subcontractors, based on their information, and is not meant to assess the concept, mathematical validity, or business validity of IOG's product. This report does not assess the implementation regarding financial viability nor suitability for any purpose.

Scope and Methodology

Tweag looks exclusively at the on-chain validation code provided by IOG. This excludes all the frontend files and any problems contained therein. Tweag manually inspected the code contained in the respective files and attempted to locate potential problems in one of these categories:

a) Unclear or wrong specifications that might allow for fringe behavior.

b) Implementation that does not abide by its specification.

c) Vulnerabilities an attacker could exploit if the code were deployed as-is, including:

- race conditions or denial-of-service attacks blocking other users from using the contract,
- incorrect dust collection and arithmetic calculations (including due to overflow or underflow),
- incorrect minting, burning, locking, and allocation of tokens,
- authorization issues,

d) General code quality comments and minor issues that are not exploitable.

Where applicable, Tweag will provide a recommendation for addressing the relevant issue.

Chapter 2

Audit

2.1 Methodology

Tweag analyzed the validator scripts comprising Djed, contained in the repository[1] as of commit `3cba2fd`[2]. The names of the files considered in this audit and their `sha256sum` are listed in Table 2.1.

Our analysis is based on the documentation provided by IOG, and on weekly meetings with IOG. The relevant documentation files are listed in Table 2.2 and their contents will be referred to as *the specification* of the protocol.

2.2 Findings

Table 2.3 lists our concerns with the current Djed implementation based on our partial exploration during a limited period of time. Throughout the rest of this section, we will detail each of our findings.

2.2.1 Vulnerabilities

2.2.1.1 Additional tokens of the order token currency symbol can be minted

Severity: *High*

When burning an order token, module `Contracts.Request` makes checks on the tokens with token name `"DjedOrderTicket"` to ensure no additional order tokens can be minted or burned. However, any other token name is unchecked. This entails that whenever an order token is burned (when it is canceled, processed, or when a successful complaint is issued), any number of tokens with the same minting policy but different token names can be minted and directed to any wallet.

In `Audit.Traces.Attacks`, traces `tryAddNewTokenWhileCanceling`, `tryAddTokenWhileProcessing`, and `tryAddTokenWhileBurning` mint such an additional token at each place where an order token is burned.

Although there are currently no apparent ways for an attacker to take advantage of these additional tokens, it is reasonable to assume that tokens of this currency should remain in orders. This might help future-proof the product if it ever starts using other token names in addition to `"DjedOrderTicket"`.

To fix this issue, we recommend providing an additional constraint on the minted value in module `Contracts.Request` that specifies that *only* tokens with the `"DjedOrderTicket"` token name are present in the minted value.

[1] `https://github.com/input-output-hk/stablecoin-plutus`
[2] Full commit hash: `3cba2fddee1849bf90817e53847e8da138580e21`

sha256sum	File Name
a57c07e...88cd374	onchain/Cardano/Djed/Contracts/ConfigManager.hs
08426af...e214c8e	onchain/Cardano/Djed/Contracts/Oracle.hs
03a193f...cbf17f4	onchain/Cardano/Djed/Contracts/Request.hs
2b3ba89...eda76b7	onchain/Cardano/Djed/Contracts/RewardFee.hs
28ef350...fbf75c3	onchain/Cardano/Djed/Contracts/StableCoin.hs
1df3781...fd7b1a9	onchain/Cardano/Djed/Core/ConfigManager.hs
6050b84...a5abc7b	onchain/Cardano/Djed/Core/Oracle.hs
6caac78...ed7ee24	onchain/Cardano/Djed/Core/OrderMinting.hs
f1917cb...51513d6	onchain/Cardano/Djed/Core/Request.hs
4086756...818a260	onchain/Cardano/Djed/Core/RewardFee.hs
a801ff5...abcf647	onchain/Cardano/Djed/Core/StableCoin.hs
c81a60d...2704b2b	onchain/Cardano/Djed/CustomContext.hs
7bc5229...971d9c9	onchain/Cardano/Djed/CustomRatio.hs
631e8ab...1bb519c	onchain/Cardano/Djed/Scripts/Oracle.hs
a2c8ded...d1617fe	onchain/Cardano/Djed/Scripts/Request.hs
31f30f4...7375cfa	onchain/Cardano/Djed/Scripts/RewardFee.hs
a9c1140...cd9a605	onchain/Cardano/Djed/Scripts/StableCoin.hs
6e49426...07ac95e	onchain/Cardano/Djed/Utils.hs

TABLE 2.1: *On-chain code source files and their `sha256sum` that were analyzed as part of the review*

sha256sum	File Name
48308d1...00f11e9	doc/ARCHITECTURE.adoc

TABLE 2.2: *Documentation files and their `sha256sum` that were used as the specification*

Severity	Section	Summary
High	2.2.1.1	Additional tokens of the order token currency symbol can be minted
High	2.2.2.1	Non-Djed tokens are not guaranteed to go back to order submitters
High	2.2.2.2	Documentation comments and in-code comments are lacking
High	2.2.2.3	There is no text in the specification for `mintSC` and `mintRC`
Medium	2.2.2.4	Minting non-Djed tokens at the same time as an order token is impossible
Medium	2.2.2.5	Requirements about sorting orders when processing them are inconsistent
Medium	2.2.2.6	There are discrepancies regarding the positions of UTxOs in outputs
Medium	2.2.2.7	Specification and implementation disagree on absence of stablecoin in input
Medium	2.2.2.8	The documentation comment in the stablecoin contract is incorrect
Medium	2.2.3.1	The onchain sorting algorithm for orders is costly
Medium	2.2.3.2	The implementation of function `processOrders` lacks clarity
Low	2.2.2.9	The oracle and stablecoin cannot be terminated simultaneously
Low	2.2.3.3	Part of the datum in the stable coin UTxO is redundant with the value
Low	2.2.3.4	Recursive operations are not implemented using a “fold” function
Low	2.2.3.5	Functions `mintSC` and `mintRC` have misleading names
Low	2.2.3.6	Interval variable names are confusing
Lowest	2.2.2.10	Specification could be improved with visual representations
Lowest	2.2.3.7	Function `validMinAdaTransfer` uses `Either` against common usage
Lowest	2.2.3.8	There are inconsistent debug messages in reward fee contract
Lowest	2.2.3.9	Some documentation comments are not detailed enough

Table 2.3: *Table of findings*

2.2.2 Unclear Specification

2.2.2.1 Non-Djed tokens are not guaranteed to go back to order submitters

Severity: High

When submitting orders, it is possible to add extra tokens that are unrelated to the Djed protocol in order UTxOs. Although order processing is unaffected, the stablecoin validator does not require to give the extra tokens back to their submitters, as illustrated by the test cases under "Try to block transactions with unrelated parasitic token" in module `Audit.Traces.Attacks`. As a consequence, the stablecoin operator may do whatever they like with these additional tokens: in our examples (namely `tryParasiticProcessOrder` in module `Audit.Traces.Attacks`), automatic balancing directs the tokens to the stablecoin operator during a successful order processing transaction.

We suggest to modify the validation of order processing to ensure additional tokens are given back to their submitters, which would probably conform to user expectations. Whether such a modification is implemented or not, we strongly suggest to make explicit which behavior is to be expected (in the specification and documentation comments) when processing orders containing tokens that are unrelated to Djed.

2.2.2.2 Documentation comments and in-code comments are lacking

Severity: High

Many helper and auxiliary functions used by the validators lack documentation comments (including general description and specification of each parameter). Sometimes in-code comments are missing as well, especially when dealing with long functions with disjoint cases or nested conditions. Many of these functions have no straightforward name or type signature that would be enough to reasonably infer their specification and the design choices taken in their implementation. Furthermore, some of these functions deal with critical aspects of Djed: for example order processing or more generally the `StableCoin` script.

We think this may pose issues when it comes to sharing and maintaining the codebase. Here is a list of functions we think would benefit from detailed documentation and inline comments:

- Module `Contracts.Request`
  - `validOrder`
  - `validTokens`
- Module `Contracts.StableCoin`
  - `validScriptOutput`
- Module `Core.StableCoin`
  - `mintSC`
  - `mintRC`
  - `applyOrder`
  - `failedOrderToDestination`
  - `succOrderToDestination`

2.2.2.3 There is no text in the specification for `mintSC` and `mintRC`

Severity: High

When processing orders, the functions `mintSC` and `mintRC` are called to compute the resulting reserve state, i.e. how many stable coins and reserve coins are in circulation after processing, etc. They also are responsible for assessing if an order is invalid due to invalid reserve, insufficient ada, etc. However, there is little corresponding text in the architecture document explaining these functions. This makes it difficult to assess their correctness, especially in conjunction with the previous issue.

We recommend explicitly outlining the intended behavior of these functions in the specification, including all potential error cases.

2.2.2.4 Minting non-Djed tokens at the same time as an order token is impossible

Severity: Medium

When minting (or burning) tokens which are unrelated to the Djed protocol in the same transaction as an order submission, the minting policy for order tokens fails with "`oneTokenInValue': more than one token in map". This behavior is not mentioned in the specification.

Trace `tryMintNewToken` in `Audit.Traces.Attacks` exhibits this issue. We suggest to either allow minting other tokens or make explicit that it is disallowed.

2.2.2.5 Requirements about sorting orders when processing them are inconsistent

Severity: Medium

The documentation comment for the order processing operation in module `Contracts.StableCoin` for redeemer `StableCoinProcess` (line 50) implies that input UTxOs must be sorted. However, our experiments showed it was indifferent to sorting. Furthermore, there is an actual sorting operation performed by the validator (see `retrieveOrders` and `insertNormalizedOrderBy` in module `Core.StableCoin`) which we discuss in concern 2.2.3.1. Besides, according to the Extended UTxO model, a transaction has a list of output UTxOs but a *set* of input UTxOs. Thus, it is strange to expect that a validator relies on how input UTxOs are sorted. We recommend modifying or removing the associated comment.

2.2.2.6 There are discrepancies regarding the positions of UTxOs in outputs

Severity: Medium

For many validators, the specification states that some output UTxO is produced with no order requirements. However, the validator often specifies that this UTxO must be the *first* UTxO produced. For example, in module `Contracts.Oracle`, line 92 (during oracle minting) states that the first output produced must be at the oracle script address, whereas the specification simply states that *some* UTxO at this address must be produced. A similar discrepancy is also present when updating the oracle (line 160) and in other validators (`configMinting`, `orderMinting`, `stableCoinContract`, `rewardFeeContract`). We recommend unifying the specification and implementation (as well as code comments) when applicable.

2.2.2.7 Specification and implementation disagree on absence of stablecoin in input

Severity: Medium

In module `Contracts.ConfigManager`, during `configManagerMinting`, line 88 checks that no stablecoin is present in the input. However, the specification makes no such requirement. As before, we recommend unifying the specification and implementation accordingly.

2.2.2.8 The documentation comment in the stablecoin contract is incorrect

Severity: Medium

In module `Contracts.StableCoin`, lines 26-27 state "It can solely be used by the stablecoin operator..." However, this comment is incorrect. A user can interact with the stablecoin contract and spend the stablecoin UTxO when submitting a complaint. We recommend editing this comment to clarify this.

2.2.2.9 The oracle and stablecoin cannot be terminated simultaneously

Severity: Low

When terminating the stablecoin (line 188 of `Contracts.StableCoin`), the validator ensures that the stablecoin is the only input. However, the specification only states that there is one stablecoin script UTxO consumed and that no request script UTxO is consumed. In particular, this means it should be possible to terminate both the stablecoin and the oracle in one transaction, provided the `oracleOperator` and `stableCoinOperator` are the same.

Trace `tryTerminateOracleStableCoin` in `Audit.Traces.Attacks` attempts to cancel both the oracle and the stablecoin simultaneously, without success. We recommend clarifying the intended behavior in the specification, as well as modifying the implementation if necessary.

2.2.2.10 Specification could be improved with visual representations

Severity: Lowest

The contract involves various exchanges of value between users and scripts. This can be confusing and difficult to keep track of, especially when it comes to the various auxiliary amounts of Lovelace that have to be taken into account (e.g "fees" and "min adas"). The specification would benefit from an annotated visual representation of the flow of these values, in the context of one or multiple examples.

Similarly, there are many different time-related variables and constants (`orderSubmissionInterval`, `minUpdateInterval`, `oracleUpdateInterval`, etc.). It would be very helpful to include a graphical illustration of a timeline showing different timestamps and intervals involved in observation updates, order submission, order processing, complaint submission, etc.

2.2.3 Code Quality

2.2.3.1 The onchain sorting algorithm for orders is costly

Severity: Medium

During validation of an order processing transaction, the input UTxOs corresponding to each order in the batch are sorted using insertion sort (see `retrieveOrders` and `insertNormalizedOrderBy` in module `Core.StableCoin`). We think this simple algorithm may be less efficient and more costly than $O(n \log(n))$ sorting algorithms.

We understand that this choice of sorting algorithm may be justified, if, for example, processing expects sorting of orders to be done offchain most of the time (see issue 2.2.2.5). Regardless, we recommend either using a different sorting algorithm or explaining why insertion sort is used.

2.2.3.2 The implementation of function `processOrders` lacks clarity

Severity: Medium

In module `Core.StableCoin`, although function `processOrders` has great documentation comments, its implementation lacks clarity, especially when it comes to the auxiliary recursive functions it involves. We think the different cases should be commented and which parameters are accumulators should be made more explicit (for now these are sandwiched between other function parameters).

2.2.3.3 Part of the datum in the stable coin UTxO is redundant with the value

Severity: Low

The bank state contains information about the current bank reserve in Lovelace (`scReserve`), and the number of stable and reserve coins in circulation (`scN_SC` and `scN_RC`).

Given that

- the stable and reserve coins are minted once and for all,
- the total amount is fixed and known,
- there is only one legitimate stablecoin UTxO associated to these specific stable and reserve coins (because of the single NFT minted at the same time), and
- order processing, fee collection, and user complaint are atomic actions in terms of transactions,

the value contained in the UTxO seems to already provide the information described by the three datum fields mentioned above. Therefore, we think there is no reason to keep these fields in the datum, increase its weight, and keep it in sync with the value.

2.2.3.4 Recursive operations are not implemented using a “fold” function

Severity: Low

The codebase has many functions involving auxiliary recursive functions that seem to be implementable using functions from the “fold” family (e.g. `foldr`). When applicable, we think this would clarify the implementation of these functions (for example, there would be no ambiguity regarding what represents the accumulator as mentioned in issue 2.2.3.2).

2.2.3.5 Functions `mintSC` and `mintRC` have misleading names

Severity: Low

In module `Core.StableCoin`, functions `mintSC` and `mintRC` do not deal with actually minting or burning tokens which makes their names misleading. Furthermore, their actual role is not made clear enough (see concerns 2.2.2.2 and 2.2.2.3). We suggest to rename these functions and detail their behavior.

2.2.3.6 Interval variable names are confusing

Severity: Low

In module `Core.Oracle`, many constants have `interval` as part of their name. However, they actually represent interval *length*. The affected constants include `submissionInterval` and `minUpdateInterval`, as well as `oracleUpdateInterval` which is a field of the oracle parameters. A similar misnomer is present in module `Core.StableCoin` with `orderInterval`. We recommend changing these variable names to alleviate this confusion.

2.2.3.7 Function `validMinAdaTransfer` uses `Either` against common usage

Severity: Lowest

In module `Core.StableCoin`, function `validMinAdaTransfer` returns a `Left` value for successful orders and a `Right` value for invalid orders. Even though none of these cases corresponds to an actual error regarding validation, only two possible outcomes of order processing, we think this goes against intuition and common usage.

2.2.3.8 There are inconsistent debug messages in reward fee contract

Severity: Lowest

In the reward fee contract (module `Contracts.RewardFee`), when there is an error, some debug messages display what the expected correct behavior is, while others display the incorrect behavior that occurred. For example, line 47 states "StableCoin NFT not expected as input !!!" (expected behavior), while line 50 states "Claiming deadline not reached yet !!!" (incorrect behavior that occurred). We recommend modifying relevant debug messages to avoid inconsistency.

2.2.3.9 Some documentation comments are not detailed enough

Severity: Lowest

Some sentences in documentation comments do not provide actual information. For example, in module `Contracts.StableCoin`, some requirements lack details. This includes but is not limited to line 66 ("Expected number of order tokens is burnt") or line 37 ("The stablecoin parameters are properly set w.r.t. the djed state and satisfy the djed parameter constraints"). We advise elaborating on these comments to provide more useful information.

2.3 Conclusion

This report outlines the 20 concerns that we have gathered while inspecting the design and code of Djed, pertaining to the code contained in the files listed in Table 2.1. As stated in Chapter 1, Tweag does not recommend for nor against the use of any work referenced in this report. Nevertheless, the existence of *high* severity concerns is a warning sign.

Appendix A

Appendix

January 24, 2023

A.1 Comments on post-audit findings by IOG

IOG has conducted their own analysis of Djed. This section describes and comments on their findings and the updates they have applied accordingly.

A.1.1 Token duplication

The audited version of Djed was vulnerable to token duplication (NFTs and order tokens) when minting two occurrences on two different transactions. This was due to an error in the implementation of the helper function `utxoConsumed'` in module **`Cardano.Djed.CustomContext`**. This function folds over input UTxOs in order to check whether a target UTxO is present. It accumulates a boolean stating whether the target UTxO is present (`True`) or not (`False`) using the OR operator (`||`). The base case for an empty list of input UTxOs is supposed to be `False` (target UTxO not in the empty list) but it had been set to `True` by mistake, resulting in a function that always returns `True` and thus never guards against transactions where the target UTxO is not present. Therefore, it was possible to mint NFTs and order tokens (one by one) without consuming the unique UTxO that would have then not been available for future transactions, hence allowing an unlimited number of such transactions, token duplication, and creating a critical vulnerability.

IOG has fixed the implementation error, changing the base result from `True` to `False`.

We agree that the fix is relevant and required. It should prevent the use of this vector of attack to duplicate tokens that must be minted once only. However, there is no absolute guarantee that token duplication is not possible in some other way that may not yet have been identified.

A.2 Audit remediation by IOG

IOG has addressed our remarks after delivery of the Djed audit report.

A.2.1 Concern 2.2.1.1

IOG rightly points out that, besides `OrderMinting`, 2.2.1.1 applies to `ConfigMinting` and `OracleMinting` as well. The concern dealt with minting policies that restrict the minting/burning of tokens of a given name but do not restrict minting/burning of tokens with other names. Since the latter are of no use and no impact on the product, this should pose no immediate security issue. Nonetheless, we suggested to strengthen the minting policy to reduce the space of possible situations that may include unforeseen weaknesses in future iterations of the product. For instance, sibling tokens with different names could start to be used in the contract at some point.

IOG decided to strenghten the minting policies for `OracleMinting` and `ConfigMinting` only. They decided not to apply a similar modification to `OrderMinting` out of performance concerns because order tokens are minted and burnt a lot during the lifecycle of Djed whereas `ConfigMinting` and `OracleMinting` are only used once on deployment and once on termination.

Designing smart contracts often involves compromising between performance and security on a case-by-case basis. In this specific case, we agree with Djed that security risks are low and that their decision is reasonable. We still advise to keep the concern in mind and re-evaluate this design choice if the context changes in the future.

A.2.2 Concern 2.2.2.1

Concern 2.2.2.1 points out that tokens which are not related to Djed (that is, tokens belonging to asset classes that are neither order tokens, stable coins, reserve coins, nor Lovelace) are not guaranteed to be given back to submitters after processing if they are locked in an order at the `Request` script.

As for 2.2.1.1, IOG considers this poses no security issue and have therefore decided not to alter the design of the product. We suggested later to prohibit, in `OrderMinting`, locking such tokens on orders. IOG advocates that this would impact the maximum number of orders that can be processed during one transaction. Instead, they decided to update the specifications to clarify the situation and postpone a possible modification when they migrate Djed to Plutus V2. We concur with IOG that this is a reasonable tradeoff. The concern was more about an unspecified behavior than a security threat.

A.2.3 Concern 2.2.2.10

Given the complexity of Djed, in particular regarding the flow of Lovelace (involving “fees” and “min adas”) and the various time constraints, we suggested to provide diagrams in the specification to help grasp the big picture and improve onboarding and maintenance.

IOG has accepted this suggestion and now provides an extensive set of diagrams to illustrate every aspect of Djed. This makes for a significant improvement over the previous architecture document.

A.2.4 Other concerns

IOG has addressed all of the other concerns. They improved the specification and documentation comments in-code when applicable and updated their code. They also provided motivation for certain design choices they decided to stick with. For instance they pointed out the negative performance impact (in size or execution) of suggestions made in 2.2.3.1 (using another sorting algorithm), 2.2.3.3 (removing information from the datum that could be deduced from the value), and 2.2.3.4 (using fold functions). None of these concerns were security related. We acknowledge these modifications and responses to the audit concerns.

A.3 Disclaimer

The present appendix is not the result of any new auditing work by Tweag on Djed. No significant code analysis nor tests have been performed on the new version of the product. Hence, the comments this appendix contains are merely high level thoughts gathered after discussing and acknowledging changes made by IOG, based on our knowledge of the version of Djed we audited back in November 2022. Although unlikely, their is no guarantee that these changes have not altered the result of the previous audit, nor introduced additional weaknesses.