# Midas Update Audit

Côme du Crest

2025-07-27

## Table of contents


- Table of contents
- Midas Audit
  - Scope
  - Context
  - Status
  - Legal Information And Disclaimer
  - Optimisations and miscellaneous

# Midas Audit

This document presents the finding of a smart contract audit conducted by Côme du Crest for Midas Protocol.

## Scope

The scope includes pull requests 57, 59, 64, 65, and 68 of `midas-apps/contracts`.

These pull requests have been merged into the branch `feat/new-audit-scope`. The last commit of the branch as of the audit is `0x0694b9a`.

## Context

This audit is the third audit I perform of this contracts codebase.

- PR 57 updates deposit and redemption vaults to add custom recipient address for the different deposit and redeem functions.
- PR 59 updates deposit and redemption vaults to add an admin function to approve deposit and redemption requests in a batch.
- PR 64 adds a custom price feed that applies APR from a starting price and update time until current block.timestamp. This allows for smooth yield distribution.
- PR 65 adds a custom field `allowance` to the admin function `addPaymentToken()` that sets the custom allowance for the added token to the given value. This `allowance` is the limit amount of a specific tokens the vault can receive / return throughout deposits / redemptions.
- PR 68 is a refactoring of the admin related roles within different vaults. The goal is to remove `mTBIL` related roles from other mTokens.

A `spec document` has been shared with me to describe the intended behaviour for the added features. Deviations from the spec document are reported.

Additionally, after PR 68 has been added, the following context has been given by the protocol team:

> Context:
>
> Right now we have a lot of mTOKEN also a lot of vaults and tokens for partners, and all of those contracts inherit our MidasAccessControlRoles contract which contains basic roles including mTBILL-related roles. Seeing all those roles may confuse some users, partners or devs that are working on the integration with our contracts. So we decided to make a clean up and remove all the mtbill roles from other products

> Requirements:
>
> 1. all products except of mtbill-related should not expose any mtbill-related roles
> 2. new contracts should be upgrade-compatible with current version

### Status

No issues have been founds, some miscellaneous comments have been raised that can be acknowledged without security concerns.

The report has been sent to the core developer.

The report has been updated with comments from the protocol team.

## Legal Information And Disclaimer

1. This report is based solely on the information provided by Midas Protocol Ltd. (the “Company”), with the assumption that the information provided is authentic, accurate, complete, and not misleading as of the date of this report. The auditor has not conducted any independent enquiries, investigations or due diligence in respect of the Company, its business or its operations.
2. Changes to the information contained in the documents, repositories and any other materials referenced in this report might affect or change the analysis and conclusions presented. The auditor is not responsible for monitoring, nor will we be aware of, any future additions, modifications, or deletions to the audited code. As such, the auditor does not assume any responsibility to update any information, content or data contained in this report following the date of its publication.
3. This report does not address, nor should it be interpreted as addressing, any regulatory, tax or legal matters, including but not limited to: tax treatment, tax consequences, levies, duties, data privacy, data protection laws, issues relating to the licensing of information technology, intellectual property, money laundering and countering the financing of terrorism, or any other legal restrictions or prohibitions. The auditor disclaims any liability for omissions or errors in the findings or conclusions presented in this report.
4. The views expressed in this report are solely my views regarding the specific issues discussed within this report. This report is not intended to be exhaustive, nor should it be construed as an assurance, guarantee or warranty that the code is free from bugs, vulnerabilities, defects or deficiencies. Different use cases may carry different risks, and integration with third-party applications may introduce additional risks.
5. This report is provided for informational purposes only and should not be used as the basis for making investment or financial decisions. This report does not constitute investment research and should not be viewed as an invitation, recommendation, solicitation or offer to subscribe for or purchase any securities, investments, products or services. The auditor is not a financial advisor, and this report does not constitute financial or investment advice.
6. The statements in this report should be considered as a whole, and no individual statement should be extracted or referenced independently.
7. To the fullest extent permitted by applicable laws, the auditor disclaims any and all other liability, whether in contract, tort, or otherwise, that may arise from this report or the use thereof.

## Optimisations and miscellaneous

This part lists minor gas/code optimizations that shouldn't make the code less readable or improve overall readability. It also lists questions about unclear code segments.

### New oracle `getRoundData()` answers with subjectively outdated `updatedAt`

The `AggregatorV3Interface` provides the function `getRoundData()` which should return among other things the value `updatedAt` describing when the data value has been updated. This allows for example price data consumers to make sure they do not use a price data that is outdated with checks like `require(block.timestamp - updatedAt <= healthyDiff)` as performed in the `DataFeed` contract.

The `CustomAggregatorV3CompatibleFeedGrowth` uses the time at which the admin provided the price data as the value for `updatedAt` even though the price was re-calculated using growth APR and evaluated at `block.timestamp`.

I would expect the returned value for `updatedAt` in `CustomAggregatorV3CompatibleFeedGrowth.getRoundData()` to be `block.timestamp`, time at which the returned data has been computed.

Using the time at which the base price has been provided by the admin forces the admin to provide a new base price / APR at least every `healthyDiff` seconds even though the calculation would still be correct which may be unnecessary.

https://github.com/midas-apps/contracts/blob/d7cdf3c688aa5ab3dca0ad86aed6bcbf0de636ea/contracts/feeds/CustomAggregatorV3CompatibleFeedGrowth.sol#L357

Response:

The code was not updated. The following comment was given:

> The issue that we were fixing with this new oracle is not really a price staleness, but fair rewards distribution, because with current manual price submission system, users that enters the market later then others with benefit more as they will have the profit earlier then others. But it still very important to receive price update from the OPS team regularly as the APR changes all the time

### Vaults requests use unexpected sender value

The redemption and deposit vaults use `recipient` as the `sender` value in requests in their internal `_redeemRequest()` and `_depositRequest()` functions:

```
    mintRequests[requestId] = Request({
        sender: recipient,  // @audit confusing
        tokenIn: tokenIn,
        status: RequestStatus.Pending,
        depositedUsdAmount: calcResult.tokenAmountInUsd,
        usdAmountWithoutFees: (calcResult.amountTokenWithoutFee *
            calcResult.tokenInRate) / 10**18,
        tokenOutRate: calcResult.tokenOutRate
    });
```

The value is correct when being consumed at the time the request is approved as it is used to determine the recipient of the tokens:

```
    function _approveRequest(
        uint256 requestId,
        uint256 newOutRate,
        bool isSafe
    ) private {
        Request memory request = mintRequests[requestId];

        ...

        mToken.mint(request.sender, amountMToken);

        totalMinted[request.sender] += amountMToken;

        ...
    }
```

However the naming of this parameter is confusing and may lead to errors if the code is later update. For example, if a feature to reimburse a rejected request is added, should the tokens be sent back to initiator of the request or the intended recipient `request.sender`?

I believe renaming this field of the `Request` struct to recipient would make sense and not cause a storage conflict.

https://github.com/midas-apps/contracts/blob/d7cdf3c688aa5ab3dca0ad86aed6bcbf0de636ea/contracts/DepositVault.sol#L480

https://github.com/midas-apps/contracts/blob/d7cdf3c688aa5ab3dca0ad86aed6bcbf0de636ea/contracts/RedemptionVault.sol#L666

Response:

The code was not updated. The following comment was given:

> Rename may cause an upgradeability issue because OZ upgrades plugin does not allow var renaming by the default and we want be able to use @custom:oz-renamed-from annotation because solidity does not support natspec inside of structs, OZ states that only workaround for now is to use @custom:oz-retyped-from on the state variable that holds the struct value, i will try that, in case if it wont work i think will be better to keep the name as it is because using "unsafeAllow" feature from OZ wont be a very secure way of performing the upgrade

### `CustomAggregatorV3CompatibleFeedGrowth.setRoundData()` allows setting round data more than once in an hour

The metaspec document about the growth oracle sates "at least 1 hour should pass since the last price update (doesn't matter if it was safe or unsafe)". This can be interpreted in two different ways. The first interpretation is that the safe function should wait at least 1 hour to be callable again even though the last call from the unsafe function. The second interpretation would be that both the safe and unsafe function have to wait 1 hour before being callable again.

The function `CustomAggregatorV3CompatibleFeedGrowth.setRoundData()` does not enforce a time delay before being callable again. As such, the code implements the first interpretation of the metaspec document, if the second interpretation is expected then code needs to be modified.

https://github.com/midas-apps/contracts/blob/d7cdf3c688aa5ab3dca0ad86aed6bcbf0de636ea/contracts/feeds/CustomAggregatorV3CompatibleFeedGrowth.sol#L205-L236

Response:

This is not an issue as the first interpretation of the metaspec document is correct.

### New deposit / redeem function will check msg.sender as well as recipient

It is stated in the metaspec document that the new deposit / redeem functions with custom recipient should enforce checks on recipient of tokens. The implementation also performs the checks on the caller of the function which may or may not be intended behaviour.

https://github.com/midas-apps/contracts/blob/d7cdf3c688aa5ab3dca0ad86aed6bcbf0de636ea/contracts/DepositVault.sol#L188

Response:

This is an intended behavior.