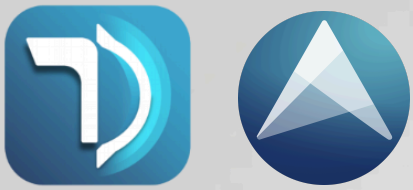

# F.T MMF AUDITING REPORT

v1.2

Jan 2025

by **Tikkala Security**
**Ancilia, Inc**

# Index

# Update History

| Revision | Description | Date |
|---|---|---|
| | | |
| | | |
| v1.2 | Final report | 02/03/2025 |
| 1.1 | Updated with change review result | 01/31/2025 |
| 1.0 | The first report | 01/16/2025 |

# Executive Summary

The Franklin Templeton team (F.T) has shared their smart contract source code in an archive. We have listed hashes of the smart contracts to ensure that the entirety of the audit can be tied to a given contract version. The Tikkala and Ancilia team collaborated with the F.T team to address all potential findings and issues. The audit scope encompassed checking for vulnerabilities in smart contracts, including re-entry attacks, logic flaws, authentication bypasses, and DoS attacks, among others.
This time, the F.T team requested an audit focused on the specific version of upgradeable contracts this time, which includes incremental changes since the last audit. Upon reviewing the codebase and documentation, we've pinpointed the significant modifications. The audit will primarily concentrate on the following files:

- MoneyMarketFund_V4.sol
- MoneyMarketFund_V5. sol
- TransactionalModule_V4.sol
- TransferAgentModule_V4.sol
- TransferAgentModule_V5.sol

Our audit efforts will be centered on those five files to ensure compliance, security, and functionality of the new changes.

# Disclaimer

Please note that security audit services cannot guarantee the discovery of all potential security issues within smart contracts. It is advisable to conduct repeated or incremental audits. Engaging multiple auditors for several audits is recommended. Product owners should maintain their own set of test cases and implement a regular code review process. Employing a threat intelligence system can aid in identifying or thwarting potential attacks, thereby reducing risk. Moreover, initiating a bug bounty program with the community can significantly enhance product security. Lastly, remember that security is complex! Even a robust smart contract does not ensure that your product is immune to all cybersecurity threats.

# Contracts overview

After compilation with Solc(version 0.8.18), there are a total of 27 smart contracts. We have listed the contract name and sha256 hash as below. The highlights are the contracts we need to focus on this time.

| Contract Name | Location | SHA256 |
|---|---|---|
| AuthorizationModule | contracts/FT/infrastructure/modules/AuthorizationModule.sol | c38939f1d8a85bef0c9cbed119e02fb94b6b64eebdd50899020f2a7c0d6ad1f5 |
| AuthorizationModuleV2 | contracts/mocks/modules/AuthorizationModuleV2.sol | 4bd1af2c037db6af680c87f3d70e539120251fda5b0e6aaa16cad9f280f74355 |
| AuthorizationModule_V1 | contracts/FT/infrastructure/modules/upgrade_history/authorization/AuthorizationModule_V1.sol | 14eff80f88d30bc3aa3e0d39c103f514cf1e2c733ef53d46eec81d02dde4f8bb |
| AuthorizationModule_V2 | contracts/FT/infrastructure/modules/upgrade_history/authorization/AuthorizationModule_V2.sol | ff6fa693f692a25013464331f7e9d203485d0f2e8914c98288a642246e9a71a9 |
| IntentValidationModule | contracts/FT/infrastructure/modules/IntentValidationModule.sol | a913e1aa692a56c15115765fb53271cf006f1a5210521bd0533c705748fd427d |
| IntentValidationModule_V1 | contracts/FT/infrastructure/modules/upgrade_history/intent_validation/IntentValidationModule_V1.sol | 587eb0c5361f8f06ff1fcb5f01b30bb74bbb3f5f4c535269e207946afbc5b983 |
| ModuleRegistry | contracts/FT/infrastructure/ModuleRegistry.sol | 55e9abfaf3cabdb38587025a0384cfa9f018ec9409ed812d9dca7e086ae799d9 |
| MoneyMarketFund | contracts/FT/MoneyMarketFund.sol | 15e5fffa77a257d53d95967598351fe99e609f3ecd175f0040bcf1ada17df493 |
| MoneyMarketFund_V1 | contracts/FT/infrastructure/modules/upgrade_history/token/MoneyMarketFund_V1.sol | b458ea1f835159119b64227449b1bb6784dcc679944bdd108a7819cd598dadeb |
| MoneyMarketFund_V2 | contracts/FT/infrastructure/modules/upgrade_history/token/MoneyMarketFund_V2.sol | 6d40acb4958a0663d2350ebc071e50a461be08873b4a1d6fc85432f3b8428dbd |
| MoneyMarketFund_V3 | contracts/FT/infrastructure/modules/upgrade_history/token/MoneyMarketFund_V3.sol | 71ece106c8f964a82ab30f9c1a8c63653ef8ff77a82a44d66bed363aede54ae8 |
| MoneyMarketFund_V4 | contracts/FT/infrastructure/modules/upgrade_history/token/MoneyMarketFund_V4.sol | 0b02ed5df407efd51573584be8043a610670d822e6208c994efcbfca18532cf1 |
| MoneyMarketFund_V5 | contracts/FT/infrastructure/modules/upgrade_history/token/MoneyMarketFund_v5.sol | b8f3def6832d1c3dd0706b4b86e74c0ef2f3f93cbfe3aa0fbcb193568ca01d39 |
| MultiSigGenVerifier | contracts/FT/infrastructure/multisig/MultiSigGenVerifier.sol | 33e5d80654df99207c9ccc883d91c6da7e41967b33d6b4901e57f44bcce59089 |
| TokenRegistry | contracts/FT/infrastructure/TokenRegistry.sol | f82eab4c3c830cea4a127d469833622fa5c4e372b6a5e69d6436b1264c4f02b6 |
| TransactionalModule | contracts/FT/infrastructure/modules/TransactionalModule.sol | d709fb07cabcbbc11144bc5b6cde6cab0eb6f2c48bb5c0965e18a9122334a5db |
| TransactionalModuleV2 | contracts/mocks/modules/TransactionalModuleV2.sol | ddbd3bf36c0392752a79956661b2f8e45e0fc914670f31cea721d630f9df7a59 |
| TransactionalModule_V1 | contracts/FT/infrastructure/modules/upgrade_history/transactional/TransactionalModule_V1.sol | cdff221d409ab73bad5b11b6d92db52ad07cf20a630fe512f51b2e7fd4b1c3c3 |
| TransactionalModule_V2 | contracts/FT/infrastructure/modules/upgrade_history/transactional/TransactionalModule_V2.sol | 37b0be70e395e3501732d22b52ab036ef5db911c8e108db07ade726a1717362d |

| Contract Name | Location | SHA256 |
|---|---|---|
| TransactionalModule_V3 | contracts/FT/infrastructure/modules/upgrade_history/transactional/TransactionalModule_V3.sol | 7bc12da1abe4d3dc55d0102c95f13a1a025ff81f9cfcac0de1b73b34e4482e62 |
| TransactionalModule_V4 | contracts/FT/infrastructure/modules/upgrade_history/transactional/TransactionalModule_V4.sol | 18dbecaaed70a077a2232520a3a9665a05a68827835afe81d199dd88b1b8ffd8 |
| TransferAgentModule | contracts/FT/infrastructure/modules/TransferAgentModule.sol | 933468ff32a6b761d67b1cf2bb2e416e2848b9613c0b828d86a81b837fb7adc6 |
| TransferAgentModule_V1 | contracts/FT/infrastructure/modules/upgrade_history/transfer_agent/TransferAgentModule_V1.sol | 514c2f6e080e61a3b5eff19cff20adecbb8010e514319acc921bffa135e70543 |
| TransferAgentModule_V2 | contracts/FT/infrastructure/modules/upgrade_history/transfer_agent/TransferAgentModule_V2.sol | 10a7e2c1b6f49080e4e8b45cef63787fb324f1a321a301b47e6c58fae09f2515 |
| TransferAgentModule_V3 | contracts/FT/infrastructure/modules/upgrade_history/transfer_agent/TransferAgentModule_V3.sol | fb8ec5c151f756d382e3b3b0ce89f67c873322d6d011ce13dedb34228421462b |
| TransferAgentModule_V4 | contracts/FT/infrastructure/modules/upgrade_history/transfer_agent/TransferAgentModule_V4.sol | 8c7116297a52cd09d75a975a80b68709ac48f236402bf5037a12f7ad231d086f |
| TransferAgentModule_V5 | contracts/FT/infrastructure/modules/upgrade_history/transfer_agent/TransferAgentModule_V5.sol | 16f743e6d082e7a772a502ef92686826b43fde25e6c07b56e9015fddf7b434b1 |

# The findings

## Results

| ID | Description | Severity | Product Impact | Status |
|---|---|---|---|---|
| **FT-A-25** | Allowing share-transfer may overpay the dividents | Critical | Critical | N/A |
| **FT-A-26** | isInstantTransferOn switch could be bypassed | Medium | Medium | Fixed |
| **FT-A-27** | CX Transfer function is Not strong | High | High | Future Improve |
| **FT-A-28** | Revert 0 amount CX Transfer | Low | Low | Fixed |
| **FT-A-29** | Use block.timestamp in EVENT | Info | Info | WON'T FIX |
| **FT-A-30** | InstantCXTransfer should check isInstantTransferOn Flag | Low | info | WON'T FIX |

## Details

### S-OFT-25 [Critical] Allowing share-transfer may overpay the dividents

The function *endofDay()* in the contract *TransferAgentModule* will process the dividend and settlements at a specific time every day.

```
209     function endOfDay(
210         address[] memory accounts⬆,
211         uint256 date⬆,
212         int256 rate⬆,
213         uint256 price⬆
214     )
215         external
216         virtual
217         override
218         onlyAdmin
219         onlyWithValidRate(rate⬆)
220         onlyValidPaginationSize(accounts⬆.length, MAX_ACCOUNT_PAGE_SIZE)
221     {
222         moneyMarketFund.updateLastKnownPrice(price⬆);
223         for (uint i = 0; i < accounts⬆.length; ) {
224             _processDividends(
225                 accounts⬆[i],
226                 moneyMarketFund.balanceOf(accounts⬆[i]),
227                 date⬆,
228                 rate⬆,
229                 price⬆
230             );
231             _processSettlements(accounts⬆[i], date⬆, price⬆);
232             unchecked {
233                 i++;
234             }
235         }
236     }
237
```

If shares can be transferred between users, dividends could be overpaid. Users can request share transfers via the *requestSelfServiceShareTransfer()* function or convince an admin to call *requestShareTransfer()* function. A *SHARE_TRANSFER* type of transaction is created and settled through the *_processSettlements()* function. User *A* transferring shares to User *B* might lead to *B* receiving extra dividends. Specifically, The dividend of *A*'s shares were calculated twice, one for *A* and one for *B*. New transaction types *CXFER_IN* could also cause this issue.

**Suggestion:** Separate the dividend and the settlement process. One at a time.

**Update:** Dev decided to fix it by using an off-chain solution. However the reviewing of the off-chain is not in the original auditing scope.

---

### S-OFT-27 [High] CX Transfer function is Not strong

When a cross-chain transfer occurs, the chain responsible for minting tokens for the user typically records the status of a unique hash generated from the transfer parameters. This helps prevent duplicate transfer transactions caused by network delays or system retries.
However, the current implementation lacks protection against such retries, which may result in duplicate transfers. Additionally, incorporating reference bytes to represent the source chain's transfer information would be beneficial.

```
367     function instantCXTransferIn(
368         address account⬆,
369         uint256 timestamp⬆,
370         uint256 amount⬆,
371         string memory memo⬆
372     )
373         external
374         virtual
375         override
376         onlyAdminOrWriteAccess
377         onlyWhenShareholderExists(account⬆)
378         accountNotFrozen(account⬆)
379     {
380         _mint(account⬆, amount⬆);
381         emit InstantCXTransferIn(account⬆, timestamp⬆, amount⬆, memo⬆);
382     }
```

**Suggestion:** Use hash to ensure the same cross chain transfer can only happen once.
**Update:** TBD. From Dev note:
"*Adding a more robust duplicate protection for transfers will be added as a day 2 improvement in the future.*"

---

### S-OFT-26 [Medium] isInstantTransferOn switch could be bypassed

The function *instantTransfer()* in contract *MoneyMarketFund_v5* enables share transfers between users and is restricted to accounts with admin or write_access roles, only when *isInstantTransferOn* is set to True.

```
251    function instantTransfer(
252        address from⬆,
253        address to⬆,
254        uint256 amount⬆,
255        string memory memo⬆
256    )
257        external
258        virtual
259        override
260        onlyAdminOrWriteAccess
261        onlyWhenShareholderExists(from⬆)
262        onlyWhenShareholderExists(to⬆)
263        accountNotFrozen(from⬆)
264        accountNotFrozen(to⬆)
265    {
266        require(isInstantTransferOn, "INSTANT_TRANSFER_CAPABILITY_NOT_ENABLED");
267        _transfer(from⬆, to⬆, amount⬆);
268        emit InstantTransfer(from⬆, to⬆, amount⬆, memo⬆);
```

However, a similar function, *transferShares()*, performs the same operation but does not require *isInstantTransferOn* to be enabled, effectively bypassing the switch check.

```
348    function transferShares(
349        address from⬆,
350        address to⬆,
351        uint256 amount⬆
352    ) external virtual override onlyAdminOrWriteAccess {
353        _transfer(from⬆, to⬆, amount⬆);
354    }
```

**Suggestion:** Disable *transferShares()*
**Update:** Fixed

---

### S-OFT-28 [Low] Revert 0 amount CX Transfer

The functions *instantCXTransferIn()* and *instantCXTransferOut()* in contract *MoneyMarketFund_v5* do not verify whether the amount is zero. Adding this check could reduce gas usage and prevent unnecessary transactions from being submitted to the blockchain.

```
367    function instantCXTransferIn(
368        address account⬆,
369        uint256 timestamp⬆,
370        uint256 amount⬆,
371        string memory memo⬆
372    )
373        external
374        virtual
375        override
376        onlyAdminOrWriteAccess
377        onlyWhenShareholderExists(account⬆)
378        accountNotFrozen(account⬆)
379    {
380        _mint(account⬆, amount⬆);
381        emit InstantCXTransferIn(account⬆, timestamp⬆, amount⬆, memo⬆);
382    }
383
        ftrace | funcSig
384    function instantCXTransferOut(
385        address account⬆,
386        uint256 amount⬆,
387        string memory memo⬆
388    )
389        external
390        virtual
391        override
392        onlyAdminOrWriteAccess
393        onlyWhenShareholderExists(account⬆)
394        accountNotFrozen(account⬆)
395    {
396        require(
397            balanceOf(account⬆) > 0 && balanceOf(account⬆) >= amount⬆,
398            "NOT_ENOUGH_BALANCE"
399        );
400        _burn(account⬆, amount⬆);
401        emit InstantCXTransferOut(account⬆, amount⬆, memo⬆);
402    }
```

**Suggestion:** revert when amount is 0

**Update:** Fixed

---

### S-OFT-30 [Low] InstantCXTransfer should check isInstantTransferOn Flag

The functions *instantCXTransferIn()* and *instantCXTransferOut()* in contract *MoneyMarketFund_v5* do not verify whether the *isInstantTransferOn* is set. Given that both function names begin with "*instant*," they should adhere to the same restriction logic as function *instantTransfer()*.

```
367     function instantCXTransferIn(
368         address account⬆,
369         uint256 timestamp⬆,
370         uint256 amount⬆,
371         string memory memo⬆
372     )
373         external
374         virtual
375         override
376         onlyAdminOrWriteAccess
377         onlyWhenShareholderExists(account⬆)
378         accountNotFrozen(account⬆)
379     {
380         _mint(account⬆, amount⬆);
381         emit InstantCXTransferIn(account⬆, timestamp⬆, amount⬆, memo⬆);
382     }
383
        ftrace | funcSig
384     function instantCXTransferOut(
385         address account⬆,
386         uint256 amount⬆,
387         string memory memo⬆
388     )
389         external
390         virtual
391         override
392         onlyAdminOrWriteAccess
393         onlyWhenShareholderExists(account⬆)
394         accountNotFrozen(account⬆)
395     {
396         require(
397             balanceOf(account⬆) > 0 && balanceOf(account⬆) >= amount⬆,
398             "NOT_ENOUGH_BALANCE"
399         );
400         _burn(account⬆, amount⬆);
401         emit InstantCXTransferOut(account⬆, amount⬆, memo⬆);
402     }
```

**Suggestion:** Check the *isInstantTransferOn* variable.

**Update:** WON'T FIX. From Dev note:

"*The instant cross-chain transfers will be done via internal app only because it has to be synchronized across two blockchains. Other instant transfers require a flag because the in some scenarios the user can directly call the ERC-20 interface in the smart contract.*"

---

### S-OFT-29 [Info] Use block.timestamp in EVENT

Several events utilize the variable date provided by the user, which can lead to confusion during off-chain parsing and make it challenging to correlate with block numbers. It is strongly recommended to include *block.timestamp* in events, such as event *DividendDistributed*, to improve clarity and traceability.

```
364     function _processDividends(
365         address account,
366         uint256 date,
367         int256 rate,
368         uint256 price
369     ) internal virtual {
370         if (moneyMarketFund.hasHoldings(account)) {
371             uint256 dividendAmount = moneyMarketFund.balanceOf(account) *
372                 uint256(abs(rate));
373             uint256 dividendShares = dividendAmount / price;
374
375             _payDividend(account, rate, dividendShares);
376             // handle very unlikely scenario if occurs
377             _handleNegativeYield(account, rate, dividendShares);
378             moneyMarketFund.removeEmptyHolderFromList(account);
379
380             emit DividendDistributed(
381                 account,
382                 date,
383                 rate,
384                 price,
385                 dividendShares
386             );
387         }
388     }
389
```

**Suggestion:** Add *block.timestamp* in.

**Update:** WON'T FIX. From Dev note:

"*The dates we provide are not full timestamps but what we consider 'Local Date', which is a date without time or zone components, it's normalized to a timezone of a specific fund.*"

---

# Summary

The Tikkala and Ancilia team conducted both automated and manual audits on the MMF smart contracts listed above. All identified issues were communicated to the F.T team via a secured channel. The audit uncovered 1 critical, 1 high, 1 medium, 2 low and 1 informational impact issues.