BlockTower/Andromeda Technical Assessment

Risk Summary/Key Takeaways

- The technical smart contract risk is considered LOW, as it will be using standard RWA contracts (formerly known as MIP21 contract) with modified Conduits that allow for swapping Dai through PSMs.
- The implementation will be made in 2 stages:
- Stage 1 will reuse a technical structure that is very similar toRWA007 and RWA014, through which Dai can be swapped to USDC through the PSM before sending the assets to the crypto broker.
- Stage 2 will feature a new set of conduit components that will allow the operator to choose with which PSM they want to interact with for each operation, effectively allowing them to choose between USDC, USDP and GUSD as target stablecoins.

- Stage 1 will reuse a technical structure that is very similar toRWA007 and RWA014, through which Dai can be swapped to USDC through the PSM before sending the assets to the crypto broker.
- Stage 2 will feature a new set of conduit components that will allow the operator to choose with which PSM they want to interact with for each operation, effectively allowing them to choose between USDC, USDP and GUSD as target stablecoins.

- Whilst technical risk is considered low, the proposed solution requires manual/human monitoring of off-chain data, which will require that there are clear processes in place for monitoring the ongoing performance of the collateral based on the reporting required by the various actors.
- Any technology required for the off-chain monitoring and uploads of data to off-chain infrastructure is not in scope for this technical assessment.
- The counterparty crypto broker infrastructure and security is out of scope of this technical assessment.

General Information

- Symbol:

RWA015

- Token Name:

RWA-015

- Ilk Registry Name:

RWA015-A: BlockTower Andromeda

- Total Supply:

1

- Relevant Information:
- MIP90: Liquid AAA Structured Credit & Money Market Fund
- Project Andromeda: Legal & Risk Assessment

- MIP90: Liquid AAA Structured Credit & Money Market Fund
- Project Andromeda: Legal & Risk Assessment

- Github Repository:

rwa-toolkit

- Collateral Type Adapter:

The collateral will use the authed join and exit functions.

Technical Information

- Implements ERC20 Token Standard:

Yes

- Compiler Version:

solidity:0.6.12

- Decimals:

18

- Overflow checks:

Yes

- Mitigation against overflow race-condition:

No

- Upgradeable contract patterns:

No

- Access control or restriction lists:

No.

- Non-standard features or behaviors:

No.

- Key addresses:
- The auth governance address for RwaLiquidationOracle

, RwaUrn

, RwaSwapOutputConduit

and RwaSwapInputConduit

contracts.

- The operator address that is permitted to operate on the RwaUrn

, RwaSwapOutputConduit

and RwaSwapInputConduit

contracts. This can be multiple addresses, however, each address must be approved by governance.

- The auth governance address for RwaLiquidationOracle

, RwaUrn

, RwaSwapOutputConduit

and RwaSwapInputConduit

contracts.

- The operator address that is permitted to operate on the RwaUrn

, RwaSwapOutputConduit

and RwaSwapInputConduit

contracts. This can be multiple addresses, however, each address must be approved by governance.

Transaction Outline

Parameter

	Value
Debt Ceiling (line):	1,280,000,000 (1.28 billion)
Stability Fee (duty):	0
Onchain Liquidation Ratio (mat):	100 %
AutoLine parameters:	Line: 1.28B, Gap: 50M, TTL: 86400 seconds (1 day)
Oracle price (pip/ds-value):	1,280,000,000
Debt write-off timelock (tau):	0
Hash of borrower’s final term sheet with MakerDAO (doc):	N/A
Recipient ETH address (kiss):	To be provided by BlockTower, and verified by Dewiz
Liquidation Process:	MIP21c3

Implementation Design Rationale

- Onboard and activate a RWA vault for the purpose of acquiring stablecoins via PSM and investing it in US Treasury Bills, held by a trust arranged and maintained by BlockTower.
- Phase 1 will consist only of the USDC PSM.
- Phase 2 will allow the operator to choose which PSM to use to obtain either USDC, USDP or GUSD.

- Phase 1 will consist only of the USDC PSM.
- Phase 2 will allow the operator to choose which PSM to use to obtain either USDC, USDP or GUSD.

Operational Security Considerations

- Dai generated through RWA vault is technically unbacked until the investment in assets has been made.
- Therefore, If Dai or stablecoin is lost by BlockTower or Galaxy/Coinbase before investing it into assets, due to hacking, operational mistakes, bugs, legal issues, or malicious activity, that would leave a deficit to be absorbed by the surplus buffer.
- As a result, Dai token holders are exposed to security, hacking, bugs, legal and/or malicious actions whilst funds are in transit.
- Given the foregoing, we would recommend that transited funds be limited to 100M or below our surplus buffer balance
- A test of 2.5M Dai will be executed to validate the entire transaction flow, before initiating incrementally drawing down the full debt ceiling.
- After that, we will enable the AutoLine

module for RWA015-A, which will gradually increase the debt ceiling for draw-downs at a predefined rate.

- Review of the crypto brokers (Galaxy and Coinbase) and their internal infrastructure and security is not possible and thus out of scope of this assessment.

Modifications in the original RWA format

Manual fee payment to the surplus buffer

The RWA vault will not have any on-chain stability fee, since it only complicates accounting as the underlying rate is variable, while the on-chain SF does not guarantee final repayment. Therefore, this setup will use RwaJar

, similar to RWA007, RWA009 and RWA014 for manual payments to the surplus buffer.

The RwaJar

contract contains a function void()

that allows the Trust on a monthly basis to pay fees (DAI sent to this contract) directly to the surplus buffer. This module uses similar logic that MakerDAO is using currently to repay core unit smart contract DAI to the Surplus Buffer.

Swap Conduits

In order to support the operational flow of swapping Dai for stablecoins through the PSMs, whilst mitigating counterparty risks of single parties carrying out manual exchange operations, for phase 1

we will leverage the RwaSwapInputConduit2

and RwaSwapOutputConduit

contracts that can automatically swap outbound Dai for stablecoins and vice versa.

For phase 2

we will implement a new version of the conduits that will allow the authorized party to chose with which PSM they want to interact before push

ing assets out of the conduits. These novel contracts will be called RwaMultiSwapInputConduit

and RwaMultiSwapOuptutConduit

.

The development of such contracts is in a very early stage, and a follow-up assessment regarding phase 2

and its operational flow will be added to this post when we reach an agreement about the design.

Overview of the Operational Flow

Setting up the vault

- Deploy contracts:
- RwaUrn
- RwaToken
- RwaSwapOutputConduit

(allows automatic swap from Dai to stablecoin before sending funds to the crypto broker, to help empty PSM, mitigate counterparty risk and avoid slippage)

- RwaSwapInputConduit2

for RwaUrn

(allows repayments in stablecoin to the RwaUrn

to be done permissionlessly)

- RwaSwapInputConduit2

for RwaJar

(allows payments in stablecoin to the RwaJar

to be done permissionlessly)

- RwaJar

(for fee payments)

- RwaUrn
- RwaToken
- RwaSwapOutputConduit

(allows automatic swap from Dai to stablecoin before sending funds to the crypto broker, to help empty PSM, mitigate counterparty risk and avoid slippage)

- RwaSwapInputConduit2

for RwaUrn

(allows repayments in stablecoin to the RwaUrn

to be done permissionlessly)

- RwaSwapInputConduit2

for RwaJar

(allows payments in stablecoin to the RwaJar

to be done permissionlessly)

- RwaJar

(for fee payments)

- Deploy executive spell which:
- Sets all risk parameters for the vault (see table in “Transaction Outline” above)
- Sets Ankura as operator

of RwaUrn

, and all Conduit contracts

- Gives MCD_PAUSE_PROXY

ward

role for RwaUrn

and all Conduit contracts

- Sets the crypto broker custody ETH addresses as whitelisted destination addresses for stablecoin (kiss

address) in the RwaSwapOutputConduit

. * This destination address can only be set or changed by Maker Governance.

- This destination address can only be set or changed by Maker Governance.
- Sets the RwaUrn

as the quitTo

address in the RwaSwapOutputConduit

- Sets the RwaUrn

as the to

address in the RwaSwapInputConduit2

for the RwaUrn

.

- Sets the crypto broker custody ETH address as the quitTo

address in the RwaSwapInputConduit2

for the RwaUrn

.

- Sets the RwaJar

as the to

address in the RwaSwapInputConduit2

for the RwaJar

.

- Sets the crypto broker custody ETH address as the quitTo

address in the RwaSwapInputConduit2

for the RwaJar

.

- Locks the RWA015 token into the RwaUrn

at an Oracle price of the debt ceiling.

- Sets an AutoLine for the RWA015-A ilk with a debt ceiling and cooldown period.
- Sets all risk parameters for the vault (see table in “Transaction Outline” above)
- Sets Ankura as operator

of RwaUrn

, and all Conduit contracts

- Gives MCD_PAUSE_PROXY

ward

role for RwaUrn

and all Conduit contracts

- Sets the crypto broker custody ETH addresses as whitelisted destination addresses for stablecoin (kiss

address) in the RwaSwapOutputConduit

. * This destination address can only be set or changed by Maker Governance.

- This destination address can only be set or changed by Maker Governance.
- Sets the RwaUrn

as the quitTo

address in the RwaSwapOutputConduit

- Sets the RwaUrn

as the to

address in the RwaSwapInputConduit2

for the RwaUrn

.

- Sets the crypto broker custody ETH address as the quitTo

address in the RwaSwapInputConduit2

for the RwaUrn

.

- Sets the RwaJar

as the to

address in the RwaSwapInputConduit2

for the RwaJar

.

- Sets the crypto broker custody ETH address as the quitTo

address in the RwaSwapInputConduit2

for the RwaJar

.

- Locks the RWA015 token into the RwaUrn

at an Oracle price of the debt ceiling.

- Sets an AutoLine for the RWA015-A ilk with a debt ceiling and cooldown period.

Drawing Dai, swapping for stablecoin and investing into bonds

[

draw-flow

1118×772 106 KB

](//makerdao-forum-backup.s3.dualstack.us-east-1.amazonaws.com/original/3X/0/c/0cf7d82e4678b7d04dd3757b2542a8e37f02c1c1.png)

- Paying agent calls draw

to generate Dai, which is sent to the RwaSwapOutputConduit

. (See step 1 in the diagram below).

- Paying agent calls pick

and specifies their custody address at the broker (Galaxy or Coinbase). They can only pick addresses whitelisted by Maker Governance.

- Paying agent calls push

in the RwaSwapOutputConduit

which swaps Dai to stablecoin, and sends it to the crypto broker custody address in a single atomic transaction. (See step 3 in diagram below)

- The paying agent exchanges stablecoin to USD on the crypto broker.
- The paying agent sends USD to the Arranger, which will invest into assets according to the agreed upon asset allocation strategy by Maker Governance.
- The Arranger provides confirmation to the Maker Community that the investment has been made.
- The debt ceiling of the vault is increased after the cooldown period specified the debt ceiling instant access module (AutoLine

).

- Repeat until the debt ceiling is maxed.

Paying vault fees/profit share

[

pay-fees-dai-flow

1026×765 82.3 KB

](//makerdao-forum-backup.s3.dualstack.us-east-1.amazonaws.com/original/3X/8/3/8368d6b9d49eeef7c46cf02900628f738c39e9fe.jpeg)

1. The paying agent sends Dai to RwaJar

contract.

1. Anyone can call void

, to transfer Dai in the RwaJar

contract to the surplus buffer

[

pay-fees-usdc-flow

1095×765 104 KB

](//makerdao-forum-backup.s3.dualstack.us-east-1.amazonaws.com/original/3X/a/1/a1030ebee80f5559b41616a053257abc44fef27e.jpeg)

1. The paying agent sends USDC to the RwaSwapInputConduit2

specific to the RwaJar

1. Anyone can call push

in RwaSwapInputConduit2

to move USDC out of it, which is swapped for Dai in the PSM, and then sent to the RwaJar

.

1. Anyone can call void

, to transfer Dai in the RwaJar

contract to the surplus buffer

Paying back principal

[

repay-principal-dai-flow

982×525 75.8 KB

](//makerdao-forum-backup.s3.dualstack.us-east-1.amazonaws.com/original/3X/4/3/43b54305fff2ab15530f3482586a517aef7155a0.jpeg)

1. The paying agent sends Dai to the RwaUrn
2. Anyone can call wipe

on the RwaUrn

, to pay back the Dai debt.

[

repay-principal-usdc-flow

1095×765 100 KB

](//makerdao-forum-backup.s3.dualstack.us-east-1.amazonaws.com/original/3X/2/9/29fb82daf0fd8c2d923ed09011e77e24a5350b6b.jpeg)

1. The paying agent sends USDC to the specific RwaSwapInputConduit2

for the RwaUrn

1. Anyone can call push

in RwaSwapInputConduit2

to move USDC out of it, which is swapped for Dai in the PSM, and then sent to the RwaUrn

.

1. Anyone can call wipe

on the RwaUrn

, to pay back the Dai debt.

Liquidations & Losses

- Monitoring of the RWA investments happens off chain.
- Maker Governance can manually trigger a liquidation event by calling the Liquidation Oracle in an executive spell.
- In case a liquidation is triggered, BlockTower will sell off assets and pay back Dai to the RwaUrn

(see paying back principal figures above).

- If the RwaUrn

has not been fully repaid after a liquidation event, Maker Governance can write off the debt to the surplus buffer using the LiquidationOracle through an executive spell.

Architecture

RWA Contracts

In order to onboard Andromeda, the standard RWA contracts will be utilized. RWA turns off automatic liquidation of the vault and ensures that the borrower is prevented from minting more DAI than the Debt Ceiling (line). Any liquidation of the vault would require that Maker Governance trigger a liquidation.

For the proposed implementation, the following RWA contracts will be utilized:

- RwaToken
- RwaUrn
- RwaLiquidationOracle
- RwaJar
- RwaSwapOutputConduit
- RwaSwapInputConduit2

As part of the spell, after the Urn has been set up, the spell will lock the ERC20 token into the RWAUrn. As this token is solely acting as a placeholder for the actual assets held by the Trust and automatic liquidation has been disabled, the on-chain value that will be reported will be the same as the debt ceiling.

Summarized below is a summary of the RWA contracts which will be utilized.

RwaToken

Source code

A standard implementation of the ERC20 token standard, with the balanceOf(address) of the deployer of the contract being set to 1 WAD at deployment. There are 18 decimals of precision.

There are three state changing functions, that are all available to the token holder, and are specific to the ERC20 token standard:

- transfer(address dst, uint wad) external returns (bool)
- transferFrom(address src, address dst, uint wad) public returns (bool)
- approve(address usr, uint wad) external returns (bool)

RwaUrn

Source code

The RwaUrn

is unique to each RWA collateral type. Aside from the core DSS wards, can, rely(address), deny(address), hope(address), and nope(address) functions, there are five functions:

- file(bytes32, address)
- lock(uint256)
- free(uint256)
- draw(uint256)
- wipe(uint256)
- exit(uint256)

The file

function can only be called by governance (via the auth modifier)

The rest of the functions can only be called by those who have been given the operator permission (hope

'd or nope

'd) on the RwaUrn

contract. And any Dai drawn by the RwaUrn

can only be sent to the RwaOutputConduit

address defined by governance when deploying the contract. In this case, the RwaOutputConduit

address will be the custody address of BlockTower.

Rwa Swap Conduits

Source code (Input Conduit)

Source code (Output Conduit)

RwaInputConduit2

and RwaOutputConduit

are two contracts aimed at handling Dai routing and automatic swapping with stablecoin using the PSM.

RwaSwapOutputConduit

has two main functions: pick(address)

and push()

.

- pick(address)

can be called by actors who have been granted the operator role (in this case, Ankura) to specify an Ethereum address that Dai should be routed to. Only addresses whitelisted by Maker Governance through kiss

can be pick

ed, ensuring that Maker Governance controls where Dai can be routed.

- The push()

function holds transitory funds, that, upon being called, are transferred to the to

address set using the pick(address)

. Note: The push()

function has been modified from standard RWA:

- push

can only be called by the operator role.

- It will swap Dai deposited in the contract to stablecoin using the PSM before sending it to the to

address.

- A quit

function has been added to allow the transfer of deposited Dai solely back to the RwaUrn

, in case the PSM is not operational, or for operational reasons where the operator of the vault decides to cancel an outgoing transaction.

RwaSwapInputConduit2

functions in a very similar manner, but lacks the pick(address)

method, as routing of Dai can only happen to the RwaUrn

j.

RwaLiquidationOracle

Source code

The RwaLiquidationOracle

contract consists of six state-changing functions (besides the usual DSS rely(address)

, deny(address)

), all protected by the auth

modifier and can only be called by governance:

- file

can be called by governance to change the vow address (used in cull

).

- init

is the initialization function. It takes 4 parameters: * ilk

: name of the vault, in this case, RWA015.

- val

: estimated value of the collateral token.

- doc

: link to legal documents representing the underlying legal scheme.

- tau

: minimum delay between the soft-liquidation and the hard-liquidation/write-off.

- ilk

: name of the vault, in this case, RWA015.

- val

: estimated value of the collateral token.

- doc

: link to legal documents representing the underlying legal scheme.

- tau

: minimum delay between the soft-liquidation and the hard-liquidation/write-off.

- bump

can be called by governance to increase or decrease the estimated value of the collateral.

- tell

can be called by governance to start a soft-liquidation.

- cure

can be called by governance after a soft-liquidation has been triggered to stop it.

- cull

can be called by governance to start a hard-liquidation/write-off. This will mark all the remaining debt of the vault as bad debt and impact the Surplus Buffer (vow).

There is one externally accessible view function called good(bytes32)

that anyone can use to check the liquidation status of the position. This function does not change contract state.

This is not a typical Maker Oracle. It will only report on the liquidation status of RwaUrn

, and can only be acted upon by governance. This oracle is not vulnerable to flash loan attacks or any manipulation aside from a governance attack.

RwaJar

Source Code

The RwaJar

contract is a permissionless contract - containing the following functions:

1. a function void()

which will transfer any DAI balance contained in the contract to the designated vow address immutably set on contract deployment.

1. A function toss(uint256 wad)

which will pull the specified amount of DAI from the sender’s wallet to the vow. This function requires that the RwaJar

contract has been previously approved by the msg.sender

to transfer the specified amount of DAI.

1. The relevant DaiJoin and Vow addresses that this contract references are established within the deployment spell - so there is no risk that the RwaJar

will send DAI to a fraudulent account address.

Contract Risk Summary

The reader should note that his assessment is solely with respect to the smart contract transactions and interfaces required to effect the on-chain state changes required under the proposed architecture and excludes any technical functionality related to the technical infrastructure required for monitoring and uploading data to MakerDAO off-chain storage and ongoing monitoring. Furthermore, this assessment does not take into consideration the operational and technical security of the counterparties BlockTower, Ankura, Galaxy or Coinbase.

Risk Analysis Conclusion: Low technical risk

The RWA code implementation resides within a sandbox-like environment, and any operation not related to locking, freeing, drawing, or wiping in the RwaUrn

and RwaJar

contracts must be voted on by governance. The code itself is lightweight. This implementation uses simplified Oracle and Urn contracts to achieve the functionality required for this specific instance of RWA.

Furthermore, most RWA contracts have been live in production for almost 2 year, and are thus deemed low risk to reuse for this implementation.

For phase 1

no novel smart contracts will be required. As mentioned throughout this assessment, we will replicate the structure already in place for previous deals.

As for phase 2

the only addition required to the RWA suite of contracts is the RWA Multi-Swap Conduits – which, as indicated, should allow swapping between any stablecoin supported through a PSM and Dai. These new contracts are currently being developed and will be reviewed before phase 2 is released.

Supporting Materials

Sūrya's Description Report

Files Description Table

File Name	SHA-1 Hash
makerdao/rwa-toolkit/src/conduits/RwaSwapInputConduit2.sol	b5e3e710fda29afea1fd0c51f1c799d42ff1c83f
makerdao/rwa-toolkit/src/conduits/RwaSwapOutputConduit.sol	d830aafa13c9ec3fba9e2bb5d6502ba40aa003e8
makerdao/rwa-toolkit/src/jars/RwaJar.sol	8fbc97752a4535ec067601e5bf1cbb1ceb505ad3
makerdao/rwa-toolkit/src/oracles/RwaLiquidationOracle.sol	88c2b4fac899d39af0198c1fb4776171e4249c19
makerdao/rwa-toolkit/src/tokens/RwaTokenFactory.sol	83ef49567342a9530fadd5f56adee7b0787a2581
makerdao/rwa-toolkit/src/tokens/RwaToken.sol	8d75732d93e0ad82a7bf3e0faf34550082291775
makerdao/rwa-toolkit/src/urns/RwaUrn2.sol	ce511f510a5d456cf686a9f64a5b46a064043c31

Contracts Description Table

Contract

Type

Bases

└

Function Name

Visibility

Mutability

Modifiers

RwaSwapInputConduit2

Implementation

└

Public

NO❗️

└

rely

External

auth

└

deny

External

auth

└

mate

External

auth

└

hate

External

auth

└

file

External

auth

└

push

External

NO❗

∟

push

External

NO❗

∟

quit

External

onlyMate

∟

quit

External

onlyMate

∟

yank

External

auth

∟

expectedDaiWad

Public

NO❗

∟

requiredGemAmt

External

NO❗

∟

_doPush

Internal

∟

_doQuit

Internal

∟

_sub

Internal

∟

_mul

Internal

RwaSwapOutputConduit

Implementation

∟

Public

NO❗

∟

rely

External

auth

∟

deny

External

auth

∟

hope

External

auth

∟

nope

External

auth

∟

mate

External

auth

∟

hate

External

auth

∟

kiss

External

auth

∟

diss

External

auth

∟

file

External

auth

∟

pick

External

NO❗

∟

push

External

onlyMate

∟

push

External

onlyMate

∟

quit

External

onlyMate

∟

quit

External

onlyMate

∟

yank

External

auth

∟

expectedGemAmt

Public

NO❗

∟

requiredDaiWad

External

NO❗

∟

_doPush

Internal

∟

_doQuit

Internal

∟

_add

Internal

∟

_sub

Internal

∟

_mul

Internal

RwaJar

Implementation

∟

Public

NO❗

∟

void

External

NO❗

∟

toss

External

NO❗

RwaLiquidationOracle

Implementation

∟

rely

External

auth

∟

deny

External

auth

∟

add

Internal

∟

mul

Internal

∟

Public

NO❗

∟

file

External

auth

∟

init

External

auth

∟

bump

External

auth

∟

tell

External

auth

∟

cure

External

auth

∟

cull

External

auth

∟

good

External

NO❗

RwaTokenFactory

Implementation

└

createRwaToken

Public

NO❗

RwaToken

Implementation

└

add

Internal

└

sub

Internal

└

Public

NO❗

└

transfer

External

NO❗

└

transferFrom

Public

NO❗

└

approve

External

NO❗

RwaUrn2

Implementation

└

Public

NO❗

└

rely

External

auth

∟

deny

External

auth

∟

hope

External

auth

∟

nope

External

auth

∟

file

External

auth

∟

lock

External

operator

∟

free

External

operator

∟

draw

External

operator

∟

wipe

External

NO!

∟

quit

External

NO!

∟

add

Internal

∟

sub

Internal

∟

mul

Internal

∟

divup

Internal

∟

rad

Internal

Legend

Symbol

Meaning

Function can modify state

Function is payable

Inheritance Graph

There are no inherited contracts in the RWA contacts (excluding tests).

[

inheritance-graph

1472×59 12.7 KB

](//makerdao-forum-backup.s3.dualstack.us-east-1.amazonaws.com/original/3X/2/4/246c1034d3c80f0c93d14b1d6dc64937f0718de1.png)

Interaction Graph

[

interaction-graph

520×1081 80 KB

](//makerdao-forum-backup.s3.dualstack.us-east-1.amazonaws.com/original/3X/b/d/bda62c8a49f3609be86a731eee4e2d1fb6eaf722.png)

Call Graph

[

call-graph

800×5756 554 KB

](//makerdao-forum-backup.s3.dualstack.us-east-
1.amazonaws.com/original/3X/7/4/748f06fc32ba8226efc1ae589d0c08be28fa59ac.jpeg)