Risk Summary/Key Takeaways

- The technical smart contract risk is considered LOW, as it will be using standard MIP21 contracts with slightly modified Conduits that allow for swapping Dai through the PSM.
- 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 offchain monitoring and uploads of data to off-chain infrastructure is not in scope for this technical assessment.
- The counterparty custodian infrastructure and security is out of scope of this technical assessment.
- CES recommends that upgrades to the MIP21 contracts and/or protocol contracts are made in the future to bring more transparency on-chain with respect to the status of the investment at any point in time. Monetalis is currently investigating a possibility to tokenize the collateral portfolio.

General Information

- Symbol:

RWA007

- Token Name:

RWA-007

- Ilk Registry Name:

RWA007-A: Monetalis Clydesdale

- Total Supply:

1

- Relevant MIP Information:
- MIP65: Monetalis Clydesdale: Liquid Bond Strategy & Execution
- MIP21: Real World Assets – Off-Chain Asset Backed Lender
- MIP65 Counterparties Community Assessment
- MIP65: Monetalis Clydesdale: Liquid Bond Strategy & Execution
- MIP21: Real World Assets – Off-Chain Asset Backed Lender
- MIP65 Counterparties Community Assessment
- Monetalis Website:

https://monetalis.io/

- Github Repository:

GitHub - makerdao/mip21-toolkit: SW Repo; Content Manager: CES-001; MIP21 Toolkit: Equipment for Off-chain Asset Backed Lending in MakerDAO

- Collateral Type Adapter:

The collateral will use the MIP21 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 and RwaConduit contracts.
- The operator address that is permitted to operate on the RwaUrn and RwaConduit contracts. This can be multiple addresses, however, each address must be approved by governance.
- The auth governance address for RwaLiquidationOracle, RwaUrn and RwaConduit contracts.
- The operator address that is permitted to operate on the RwaUrn and RwaConduit contracts. This can be multiple addresses, however, each address must be approved by governance.

Transaction Outline

Parameter

Value

Debt Ceiling (line):

250,000,000 (250 Million)

Stability Fee (duty):

0

Onchain Liquidation Ratio (mat):

100 %

AutoLine parameters:

Line: 250M, Gap: 50M, TTL: 604800 seconds (1 week)

Oracle price (pip/ds-value):

250,000,000

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 Monetalis, and verified by CES

Liquidation Process:

MIP21c3

Implementation Design Rationale

- MIP65 aims to onboard and activate a RWA vault for the purpose of acquiring USDC via PSM and investing it in high quality liquid bond strategies held by a trust arranged and maintained by Monetalis.

Operational Security Considerations

- Dai generated through RWA vault is technically unbacked until investment into assets has been made.
- Therefore, If Dai or USDC is lost by Monetalis or 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.
- Therefore it is recommended that the Trustee (James Asset (PTC) Limited itself managed by Riverfront Capital Ltd) receive and manage their funds through a professional custody provider. Each operation should be validated by an independent third party, Hatstone.
- Furthermore CES recommends that the debt ceiling is drawn down gradually in increments which ensures the amount of “funds in transit” is always below the surplus buffer in the worst case scenario where the USDC is lost before being invested. Therefore a debt ceiling instant access module to gradually increase the debt ceiling to achieve the gradual drawdown will be utilized.
- A test of 1M Dai will be executed to validate the entire transaction flow, before initiating incrementally drawing down the full 250M Dai.
- Review of the custodian (Coinbase) and its internal infrastructure and security is not possible and thus out of scope of this assessment.

Modifications to standard MIP21

Manual repayments to surplus buffer

The RWA vault will not have any onchain stability fee, since it only complicates accounting as the underlying rate is variable, while the onchain SF does not guarantee final repayment. Therefore this setup will use RwaJar, similar to RWA-009 for manual payments to the surplus buffer.

The RwaJar contract contains a function void()

that allows the Trust on a monthly basis to repay fee/interest remittances (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.

USDC Swap Conduits

In order to support the operational flow of swapping Dai for USDC through the PSM, whilst mitigating counterparty risks of single parties carrying out manual exchange operations, a new Input and OutputConduit will be implemented that automatically can swap outbound Dai for USDC and vice versa.

Overview of the Operational Flow

Setting up the vault

- Deploy contracts:
- RwaUrn
- RwaToken
- RwaSwapOutputConduit (allows automatic swap from Dai to USDC before sending funds to Coinbase, to help empty PSM, mitigate counterparty risk and avoid slippage)
- RwaSwapInputConduit for RwaUrn (allows repayments in USDC to the RwaUrn)
- RwaSwapInputConduit for RwaJar (allows repayments in USDC to the RwaJar)

- RwaJar (for fee repayments)
- RwaUrn
- RwaToken
- RwaSwapOutputConduit (allows automatic swap from Dai to USDC before sending funds to Coinbase, to help empty PSM, mitigate counterparty risk and avoid slippage)
- RwaSwapInputConduit for RwaUrn (allows repayments in USDC to the RwaUrn)
- RwaSwapInputConduit for RwaJar (allows repayments in USDC to the RwaJar)
- RwaJar (for fee repayments)
- Deploy executive spell which:
- Sets all risk parameters for the vault (see table in “Transaction Outline” above)
- Sets Monetalis as operator

of RwaUrn, and all Conduit contracts

- Sets DsPauseProxy as auth

role for RwaUrn and all Conduit contracts

- Sets the Coinbase custody ETH address as the whitelisted destination address for USDC (kiss

address) in the RwaSwapOutputConduit. 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 RwaSwapInputConduit for the RwaUrn.

- Sets the the Coinbase custody ETH address as the quitTo

address in the RwaSwapInputConduit for the RwaUrn.

- Sets the RwaJar as the to

address in the RwaSwapInputConduit for the RwaJar.

- Sets the the Coinbase custody ETH address as the quitTo

address in the RwaSwapInputConduit for the RwaJar.

- Locks the RWA007 token into the RwaUrn at an Oracle price of 250M USD.
- Sets an AutoLine for the RWA007-A ilk with a debt ceiling and cooldown period.
- Sets all risk parameters for the vault (see table in “Transaction Outline” above)
- Sets Monetalis as operator

of RwaUrn, and all Conduit contracts

- Sets DsPauseProxy as auth

role for RwaUrn and all Conduit contracts

- Sets the Coinbase custody ETH address as the whitelisted destination address for USDC (kiss

address) in the RwaSwapOutputConduit. 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 RwaSwapInputConduit for the RwaUrn.

- Sets the the Coinbase custody ETH address as the quitTo

address in the RwaSwapInputConduit for the RwaUrn.

- Sets the RwaJar as the to

address in the RwaSwapInputConduit for the RwaJar.

- Sets the the Coinbase custody ETH address as the quitTo

address in the RwaSwapInputConduit for the RwaJar.

- Locks the RWA007 token into the RwaUrn at an Oracle price of 250M USD.
- Sets an AutoLine for the RWA007-A ilk with a debt ceiling and cooldown period.

Drawing Dai, swapping for USDC and investing into bonds

- Monetalis calls draw

to generate Dai, which is sent to the RwaSwapOutputConduit. (See step 1 in Diagram below).

- Monetalis calls pick

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

- Monetalis calls push

in the RwaSwapOutputConduit which swaps Dai to USDC, and sends it to the Coinbase custody address in a single atomic transaction. (See step 2 in diagram below)

- The trustee exchanges USDC to USD on Coinbase.
- The trustee sends USD to Bank Sygnum and invests into assets according to the agreed upon asset allocation strategy by Maker Governance.
- Monetalis provides confirmation to the Maker Community that the investment into bonds has been made.
- The debt ceiling of the vault is increased after the cooldown period specified the debt ceiling instant access module (AutoLine).
- If Monetalis fails to confirm that funds are being correctly invested, Maker Governance can set DC to 0 until further notice.
- Repeat until the debt ceiling is maxed.

[

1356×762 52.3 KB

](//makerdao-forum-backup.s3.dualstack.us-east-1.amazonaws.com/original/3X/7/e/7e5a05130ca5d963c02abade35d4394babe76d9f.png)

Paying vault fees/profit share

- The trustee sends Dai to RwaJar contract. The Trustee can also use a RwaSwapInputConduit specific to the RwaJar to send USDC that is then automatically swapped for Dai in the PSM, and then sent to the RwaJar.
- Anyone can call void

, to transfer Dai in the RwaJar contract to the surplus buffer

[

1322×722 29.6 KB

](//makerdao-forum-backup.s3.dualstack.us-east-1.amazonaws.com/original/3X/8/6/8619397b5c7af5610964e6f3f50572f804a5803e.png)

[

1326×802 45.4 KB

](//makerdao-forum-backup.s3.dualstack.us-east-1.amazonaws.com/original/3X/0/7/0773fa98f100ef88d1e3437b320884f8fba06b94.png)

Paying back principal

- The Trustee sends Dai to the RwaUrn (A specific RwaSwapInputConduit for the RwaUrn can again be used if they wish to repay using USDC to avoid slippage).
- Anyone can call wipe

on the RwaUrn, to pay back the Dai debt.

[

1322×576 26.3 KB

](//makerdao-forum-backup.s3.dualstack.us-east-1.amazonaws.com/original/3X/8/9/89dfce9205016f8c28e07611f04a6034818735cf.png)

[

1326×802 42.6 KB

](//makerdao-forum-backup.s3.dualstack.us-east-1.amazonaws.com/original/3X/f/7/f7f639790f679a71ebcf76858c45a1661d4f2004.png)

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 Monetalis 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.

Emergency Shutdown

- In case of Emergency Shutdown, the Trustee will start selling off the off chain assets for USDC and transfer USDC to a new RwaCageSettlement contract.
- Dai holders will be able to redeem Dai for RWA007 tokens through the End module.
- Subsequently RWA007 token holders can swap RWA007 tokens for a proportionate amount of USDC in the RwaCageSettlement contract.
- The RwaCageSettlement is still in development, but very close to being completed. When this contract is finished, it can also be reused for other RWA vaults.
- The fact that this contract is not live in production yet is not a blocker to initiate the vault, as existing RWA also lacks this easy onchain redeem functionality in case of ES. The plan is to deploy this contract for MIP65 as soon as it is ready.

[

1282×442 32.5 KB

](//makerdao-forum-backup.s3.dualstack.us-east-1.amazonaws.com/original/3X/1/1/119235a1d789801265a906710a3f6e70962f79a3.png)

Architecture

MIP21 Contracts

In order to onboard MIP65 Clydesdale the MIP21 standard contracts will be utilized. MIP21 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 MIP21 contracts will be utilized:

- RwaToken
- RwaUrn
- RwaLiquidationOracle
- RwaJar
- RwaSwapOutputConduit - new contract that integrates with the PSM to swap outbound Dai for USDC.
- RwaSwapInputConduit - new contract that integrates with the PSM to swap inbound USDC for Dai.

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 debt ceiling of 250M Dai.

Summarized below is a summary of the MIP21 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 MIP21 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 Monetalis.

RwaSwapConduits

Source code (InputConduit)

Source code (OutputConduit)

The RwaInputConduit and RwaOutputConduit are two contracts aimed at handling Dai routing and automatic swapping with USDC 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 Monetalis) 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 MIP21:

- push

can only be called by the operator role.

- It will swap Dai deposited in the contract to USDC 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.

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

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

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, RWA007.

- 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, RWA007.

- 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 offchain storage and ongoing monitoring. Furthermore, this assessment does not take into consideration the operational and technical security of the counterparties Monetalis 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, MIP21 contracts have been live in production for over a year, and are thus deemed low risk to reuse for this implementation.

The only addition required to the MIP21 suite of contracts is the addition of the new RwaSwapConduits – which as indicated is to allow automatic swapping between USDC and Dai. These new contracts are currently being reviewed internally by Protocol Engineering.

Supporting Materials

Sūrya's Description Report

Files Description Table

File Name

SHA-1 Hash

@makerdao/mip21-toolkit/src/tokens/RwaToken.sol

8d75732d93e0ad82a7bf3e0faf34550082291775

@makerdao/mip21-toolkit/src/urns/RwaUrn2.sol

ce511f510a5d456cf686a9f64a5b46a064043c31

@makerdao/mip21-toolkit/src/conduits/RwaOutputConduit3.sol

911defea762e4e5e2940c719c06b29e65059e849

@makerdao/mip21-toolkit/src/conduits/RwaInputConduit3.sol

1c2137b4193893a2770519ed4e914f9db766667c

@makerdao/mip21-toolkit/src/oracles/RwaLiquidationOracle.sol

88c2b4fac899d39af0198c1fb4776171e4249c19

@makerdao/mip21-toolkit/src/jars/RwaJar.sol

8fbc97752a4535ec067601e5bf1cbb1ceb505ad3

Legend

Symbol

Meaning

Function can modify state

Function is payable

Contracts Description Table

Contract

Type

Bases

└

Function Name

Visibility

Mutability

Modifiers

RwaToken

└

add

Internal

└

sub

Internal

└

Constructor

Public

NO❗

└

transfer

External

NO❗

└

transferFrom

Public

NO❗

└

approve

External

NO❗

RwaUrn2

└

Constructor

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

RwaOutputConduit3

└

Constructor

Public

NO!

└

rely

External

auth

└

deny

External

auth

└

mate

External

auth

└

hate

External

auth

└

hope

External

auth

└

nope

External

auth

└

kiss

External

auth

└

diss

External

auth

└

file

External

auth

└

pick

External

isMate

└

push

External

isMate

└

push

External

isMate

└

quit

External

isMate

└

quit

External

isMate

 ㄴ

quitGem

External

auth

 ㄴ

daiToGem

Public

NO!

 ㄴ

_doPush

Internal

 ㄴ

_doQuit

Internal

 ㄴ

add

Internal

 ㄴ

sub

Internal

 ㄴ

mul

Internal

RwaInputConduit3

 ㄴ

Constructor

Public

NO!

 ㄴ

rely

External

auth

 ㄴ

deny

External

auth

└

mate

External

auth

└

hate

External

auth

└

file

External

auth

└

push

External

isMate

└

push

External

isMate

└

quit

External

isMate

└

quit

External

isMate

└

quitDai

External

auth

└

gemToDai

External

NO!

└

_doPush

Internal

└

_doQuit

Internal

└

sub

Internal

└

mul

Internal

RwaLiquidationOracle

└

rely

External

auth

└

deny

External

auth

└

add

Internal

└

mul

Internal

└

Constructor

Public

NO!

└

file

External

auth

└

init

External

auth

└

bump

External

auth

└

tell

External

auth

└

cure

External

auth

└

cull

External

auth

└

good

External

NO!

RwaJar

└

Constructor

Public

NO!

└

void

External

NO!

└

toss

External

NO!

Inheritance Graph

There are no inherited contracts in the MIP21 contacts (excluding tests).

[

3896×183 25.6 KB

](//makerdao-forum-backup.s3.dualstack.us-east-1.amazonaws.com/original/3X/2/b/2b7c43ce01a82800d7bd244a2dd1fe2ceff79ee0.png)

Call Graph

[

900×6935 458 KB

](//makerdao-forum-backup.s3.dualstack.us-east-1.amazonaws.com/original/3X/4/c/4cf7e5bee92438f9163e088614601f69c3873c6c.jpeg)

Interaction Graph

[

900×1684 273 KB

](//makerdao-forum-backup.s3.dualstack.us-east-1.amazonaws.com/original/3X/e/d/ed9c3bb930e165cb6f4aecec5303db85bbeb551b.jpeg)