Security Review

Cantina Managed

SKY: ARBITRUM ONE AND BASE PARALLEL PAU CONTROLLER DEPLOYMENT VERIFICATION AND ARCHITECTURE REVIEW

Cantina Managed review by:
M4rio.eth, Lead Security Researcher
Xmxanuel, Lead Security Researcher

CANTINA

October 1, 2026

Contents

1 Introduction **2**
- 1.1 About Cantina . . . 2
- 1.2 Disclaimer . . . 2
- 1.3 Risk assessment . . . 2
  - 1.3.1 Severity Classification . . . 2

2 Security Review Summary **3**

3 Deployment Verification **4**
- 3.1 Arbitrum One and Base parallel PAU Controller: deployment attestation . . . 4

4 Parallel Controller Architecture Validation **13**
- 4.1 Summary . . . 13
- 4.2 Why a second controller . . . 13
- 4.3 How the two controllers work . . . 13
- 4.4 Risks . . . 14
- 4.5 Adding or retiring functionality . . . 16
- 4.6 Assessment . . . 17
- 4.7 Final statement . . . 18
- 4.8 Exception . . . 18

1 Introduction

1.1 About Cantina

Cantina is a security platform that pairs a world-class network of security researchers with agentic AI to help teams building onchain find and fix real risk. Through managed reviews, competitions, and bounties, Cantina secures protocols before vulnerabilities can be exploited.
Learn more at cantina.security

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity level	Impact: High	Impact: Medium	Impact: Low
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings are a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

From Sep 23nd to Sep 25th the Cantina team conducted a deployment verification and architecture review of pau controller in parallel on Arbitrum One and Base.

3 Deployment Verification

3.1 Arbitrum One and Base parallel PAU Controller: deployment attestation

On Arbitrum One and on Base, Spark has deployed a second controller for its existing `ALMProxy`. The legacy `ForeignController` stays as it is. Next to it sits a new Diamond PAU `Controller` with one facet, `CCTPFacet`, intended to burn USDC from the proxy for minting to Spark's mainnet `ALMProxy` on Ethereum over Circle's CCTP V2. The new controller comes with its own `AccessControls`, `RateLimits`, `Beacon` and an `AdministeredAgent` through which the relayer multisig calls it.

This attestation confirms that the 18 address entries added to the registry in PR #115 identify 16 new contracts, built from the pinned audited sources and wired as the PR describes, plus the two pre-existing grantor Safes. At the configuration snapshots, the chain's governance executor was the only admin of the new contracts, and the deployer retained no role on the checked contracts.

At the configuration snapshots below, the new controller could not move proxy funds: the spell that grants it access to the proxy, sets the Ethereum recipient and sets its rate limits had not run.

Configuration snapshots: Arbitrum block 507725257 and Base block 51638244, both dated 2026-09-22 UTC and re-read on 2026-09-23. A later bytecode-only check at blocks 507756092 and 51642109 confirmed unchanged runtimes.

Scope. This attests the deployed state at the configuration snapshot blocks. It does not cover the activation spell, its limits, or the combined legacy-plus-PAU exposure.

Pins. Registry: PR #115 head `900e860`. Deployment scripts: spark-pau-deploy `8982d64`, the master commit after PR #8 (Arbitrum) and PR #10 (Base). Sources: diamond-pau v1.14.0 `cbf71b2` and pau-administered-agent v1.0.0 `bfaaf70`. Compiler: solc 0.8.34, optimizer 200 runs, cancun. The Arbitrum deployment logic and inputs are unchanged from `4f724ba`, the commit the Arbitrum deployment ran from.

Bytecode. All 16 runtimes match the build. Beacon, AdministeredAgentFactory, AccessControls, RateLimits and AdministeredAgent have no immutables, and their on-chain code hash equals the artifact's `deployedBytecode` hash, metadata included. PAUFactory, Controller and CCTPFacet carry immutables. Their runtimes match outside the compiler-declared immutable slots, and every slot contains the intended constructor value on both chains. Inserting those values into the build artifacts reproduces the on-chain code hashes exactly. All eight per chain are plain CREATE deployments. All 16 are verified on Arbiscan and Basescan with these settings.

Verdict on the deployments. Arbitrum One and Base: both deployments match the PR and are correctly configured for their pre-spell state. Neither verdict extends to the activation spell or to the parallel-controller architecture. The architecture is covered in the review report; the activation spell remains outside scope.

Addresses:

Constant	Arbitrum One	Base
SPARK_BEACON	0x86036CE5d2f792367C0A A43164e688d13c5A60A8	0x7ac96180C4d6b2A328D3 a19ac059D0E7Fc3C6d41
SPARK_PAU_FACTORY	0x3968a022D955Bbb7927c c011A48601B65a33F346	0x011A115b5498B85b3d12 245A3a7296F77325B5C3
SPARK_ADMINISTERED_- AGENT_FACTORY	0xCBA0C0a2a0B6Bb11233e c4EA85C5bFfea33e724d	0xD711DbfD937a45e2C89C A0D4781cfa5BAb32e752
CCTP_FACET	0xeCCA0D296Cb133081d41 E9772B60D57F5fd2798E	0xb22d50c393c6E1D13E3e 05B172448dD8BF8DdC32
PAU_ACCESS_CONTROLS	0x8386f819860D54B11805 39Ff4852E4CAECef8A1D	0xE593c8c6a31d88cab100 244Afd352efb127f9a16
PAU_ADMINISTERED_AGENT	0x0745aae633E8318a063D 383791bCc0d8C82F46C6	0x70E46bAf2E3F27a11975 7D7b796c641C8bc087cE
PAU_CONTROLLER	0x04ACB9e9bbd64A425677 edC535D6B30cfD74E42f	0xd864bF1Ea2f78Dc2013E 3FC7e4C474383BE9d456

PAU_RATELIMITS	0x4824C4336a1a11979068A544958dCe5D49B42752	0x5E311e8BCe95F4e8d4920E70985ED4aC122a838A
PAU_GRANTOR_MULTISIG (pre-existing Safe)	0x4B61A0E48dd1e300f64090C60F414c1aC6CbC514	same

Note the cross-chain collisions: Arbitrum's SPARK_BEACON, SPARK_PAU_FACTORY, agent factory and CCTP_FACET addresses all hold unrelated contracts on Base (the first is Base's legacy ALM_CONTROLLER). Always read an address with its chain.

Bindings (identical pattern on both chains):

- `PAU_CONTROLLER.beacon()` = SPARK_BEACON, `.accessControls()` = PAU_ACCESS_CONTROLS, `.rateLimits()` = PAU_RATELIMITS, `.proxy()` = the existing legacy ALM_PROXY.
- `SPARK_PAU_FACTORY.beacon()` = SPARK_BEACON.
- `CCTP_FACET.cctp()` = Circle TokenMessengerV2 (same on both chains), `.usdc()` = native USDC.
- Admin on the new PAU contracts is SPARK_EXECUTOR, a spark-gov-relay Executor with `delay() == 0`.

Reference	Arbitrum One	Base
SPARK_EXECUTOR	0x65d946e533748A998B1f0E430803e39A6388f7a1	0xF93B7122450A50AF3e5A76E1d546e95Ac1d0F579
ALM_PROXY (existing)	0x92afd6F2385a90e44da3a8B60fe36f6cBe1D8709	0x2917956eFF0B5eaF030abDB4EF4296DF775009cA
Legacy controller	0xC40611AC4Fff8572Dc5F02A238176edCF15Ea7ba	0x86036CE5d2f792367C0AA43164e688d13c5A60A8
Legacy RateLimits	0x19d08879851fb54c2dcc4bb32b5a1ea5e9ad6838	0x983ec82e45c61a42fdda7b3c43b8c767004c8a74
Circle TokenMessengerV1	0x19330d10d9cc8751218eaf51e8885d058642e08a	0x1682ae6375c4e4a97e4b583bc394c861a46d8962
Circle TokenMessengerV2	0x28b5a0e9C621a5BadaA536219b3a228C8168cf5d	same
Native USDC	0xaf88d065e77c8cC2239327C5EDb3A432268e5831	0x833589fCD6eDb6E08f4c7C32D4f71b54bdA02913
Sky mainnet PAU Beacon	0x829dC2b7E94B1954F0764E573f2E0d45Afa28199	same
Ethereum ALM_PROXY (legacy configured, PAU planned)	0x1601843c5E9bC251A3272907010AFa41Fa18347E	same
Deployer EOA	0xC758519Ace14E884fdbA9ccE25F2DbE81b7e136f	same

The controller has no owner of its own. Its roles live in AccessControls. The factories hold no roles or further configuration.

Integration: Each Beacon and each controller holds exactly one integration, `bytes32("CCTP_FACET")` (`0x434354505f4641434554000000000000000000000000000000000000000000000000`), pointing at that chain's facet with these ten selector pairs, identical as a set to those on the Sky mainnet PAU Beacon:

Controller call	Selector	Facet delegate
`cctp_setDomainParameters(uint32,bytes32,uint32,uint32)`	0x6969ad0b	0x0f32bb10
`cctp_transfer(uint256,uint32,uint64)`	0xd82592c1	0xb9bbdcf2
`cctp_toCCTPRateLimitKey()`	0x9bbb03f2	0x127febbd
`cctp_getDomainParameters(uint32)`	0x3c7020e9	0xa24e0ef5

`cctp_getToDomainRateLimitKey(uint32)`	0xb2a4d06f	0x845da79b
`cctp_VERSION()`	0x87bd715a	0xffa1ad74
`cctp_DESTINATION_CALLER()`	0x2c7389a0	0x62cef4b9
`cctp_MIN_FINALITY_THRESHOLD()`	0x75ef4015	0x5bd0f214
`cctp_cctp()`	0xf9b8d11c	0xe3329e32
`cctp_usdc()`	0x3520c914	0x3e413bee

The controller's synced config equals its beacon's byte-for-byte, and ABI argument and return shapes match for all ten pairs. Beacon protection covers all eleven native controller selectors and no wire collides with them. All ten raw delegate selectors are unwired (facet `0x0`, delegateSelector `0x00000000`). Truncated static arguments revert, trailing calldata does not create a second dispatch, and the controller fallback bubbles the exact returndata and revert status, so failed external calls revert atomically. Integration setup rejects zero or no-code facets and empty wire arrays. The facet entries are nonpayable, so no call value can enter their function bodies.

Roles:

Contract	Role	Holders
SPARK_BEACON	DEFAULT_ADMIN_ROLE	SPARK_EXECUTOR only
PAU_ACCESS_CONTROLS	DEFAULT_ADMIN_ROLE	SPARK_EXECUTOR only
PAU_ACCESS_CONTROLS	ALLOCATOR_ROLE	PAU_ADMINISTERED_AGENT only
PAU_RATELIMITS	DEFAULT_ADMIN_ROLE	SPARK_EXECUTOR
PAU_RATELIMITS	CONTROLLER	PAU_CONTROLLER
PAU_ADMINISTERED_AGENT	admin / actor / grantor / revoker	SPARK_EXECUTOR / ALM_RELAYER_MULTISIG / PAU_GRANTOR_MULTISIG / ALM_FREEZER_MULTISIG, one each

Every AccessControl role admin is DEFAULT_ADMIN_ROLE. The deployer 0xC758519Ace14E884fdbA9ccE25F2DbE81b7e136f, both factories, the backstop relayer and the zero address hold no role on the new contracts and no agent seat. The agent has no configuration beyond its four membership sets: no rate limit, target binding, selector allowlist, expiry, delay or pause. PAU_RATELIMITS is not enumerable, so its two rows rest on role-event history checked against `hasRole`, not a member count. Role ids: `ALLOCATOR_ROLE` = `0x68bf109b95a5c15fb2bb99041323c27d15f8675e11bf7420a1cd6ad64c394f46`, `CONTROLLER` = `0x70546d1c92f8c2132ae23a23f5177aa8526356051c7510df99f50e012d221529`.

Safes (identical on both chains):

Safe	Threshold	Notes
ALM_RELAYER_MULTISIG (agent actor, legacy RELAYER)	1 of 2	either key alone authorizes relayer calls
ALM_BACKSTOP_RELAYER_MULTISIG (legacy RELAYER)	2 of 5	not a PAU actor at the snapshots
ALM_FREEZER_MULTISIG (agent revoker, legacy FREEZER)	2 of 4	
PAU_GRANTOR_MULTISIG (agent grantor)	3 of 5	

The relayer Safe is 1 of 2, so either key alone can authorize fund movements on the legacy controller and, after activation, the PAU controller, subject to each controller's limits. One owner key sits in all four Safes. Grantor and freezer share three owners, which meets both thresholds, so the same three keys can remove and re-add PAU actors. All four are Safe v1.4.1 proxies with the expected singleton and CompatibilityFallbackHandler `0xfd0732Dc9E303f09fCEf3a7388Ad10A83459Ec99`, EOA owners only, no modules.

Pre-spell state (both chains):

- Legacy proxy: SPARK_EXECUTOR admin, legacy controller holds CONTROLLER, PAU controller does not. Legacy RELAYER stays on the relayer and backstop Safes, FREEZER on the freezer Safe, and legacy `mintRecipients(0)` is the Ethereum ALM_PROXY 0x1601843c5E9bC251A3272907010AFa41Fa18347E on both chains.
- All PAU rate limits are zero and domain 0 is unconfigured on both controllers, so authorized `cctp_transfer` calls revert at the zero limits.
- Facet constants: DESTINATION_CALLER `bytes32(0)`, so anyone may relay an attested message while the recipient stays fixed. MIN_FINALITY_THRESHOLD 2000, standard finality. VERSION "1.0.0". Amount, destination domain and `feeCapRate` are per-call inputs, not stored configuration.
- Legacy CCTP still live on V1: domain-0 limit 50m / 25m per day on Arbitrum, 200m / 500m per day on Base, aggregate unlimited (table below).
- USDC balance of both proxies 0. PAU controller and agent also 0. No allowances to either messenger. All eight new contracts hold zero native ETH.

Legacy CCTP rate limits: The legacy RateLimits uses the same total and domain-0 keys as PAU but separate storage. `U` is `2^256 - 1`, the unlimited sentinel, not a balance. Amounts are raw USDC units (6 decimals), `slope` is units/second, `lastUpdated` a Unix timestamp, and `currentLimit` the getter result at the locked block.

Key	maxAmount	slope	lastAmount	lastUpdated	currentLimit
Arbitrum / total	U	0	U	1740409709	U
Arbitrum / domain 0	50000000000000	289351851	49939116590647	1790026252	50000000000000
Base / total	U	0	U	1731938717	U
Base / domain 0	200000000000000	5787037037	199831238962897	1790064573	200000000000000

Capacity towards Ethereum (Circle domain 0) was 50m USDC on Arbitrum and 200m on Base, refilling at 25m and 500m per day. That is headroom, not inventory: both proxies held zero liquid USDC at these blocks. This says nothing about their invested assets or total portfolio value.

Per-contract verification: One subsection per contract, one row per configuration surface, expected value beside observed value. Observed state comes from chain reads; runtime identity and source-level properties were checked against the pinned build and source. Values are identical on both chains unless a column shows otherwise. "Only" means the enumerable count is 1; PAU_RATELIMITS uses the role-event checks described above.

SPARK_BEACON (diamond-pau `Beacon.sol`):

Surface	Expected	Arbitrum	Base
Runtime code	build of cbf71b2, no immutables	exact, 6090 bytes	exact
`getRoleMemberCount(DEFAULT_ADMIN_ROLE)`	1	1	1
`getRoleMember(DEFAULT_ADMIN_ROLE, 0)`	SPARK_EXECUTOR	0x65d946e533748A998B1f0E430803e39A6388f7a1	0xF93B7122450A50AF3e5A76E1d546e95Ac1d0F579
`hasRole(DEFAULT_ADMIN_ROLE, deployer)`	false	false	false
`integrations().length`	1	1	1

`integrations()[0].id`	bytes32("CCTP_FACET")	match	match
`getConfig(CCTP_FACET).facet`	this chain's CCTP_FACET	0xeCCA0D296Cb133081d41E9772B60D57F5fd2798E	0xb22d50c393c6E1D13E3e05B172448dD8BF8DdC32
`getConfig(CCTP_FACET).wires`	the 10 pairs of the Sky mainnet Beacon, compared as a set	10, identical as a set	10, identical as a set
Reentrancy guard	NOT_ENTERED (1)	1	1

SPARK_PAU_FACTORY (`PAUFactory.sol`):

Surface	Expected	Observed
Runtime code	build with verified Beacon immutable	exact match (0x374aea7bebbec3447f76e262763ef0cd5b7db2e27de5417b5fcea61f88de24b5 Arbitrum, 0xf2a6a9ec61c3ca10ac9a631073161430faf6f17161dc062b07a07ed78ebb9bec Base)
`beacon()` (immutable)	SPARK_BEACON	match on both chains
Roles / owner	none exist	n/a

SPARK_ADMINISTERED_AGENT_FACTORY (`AdministeredAgentFactory.sol`):

Surface	Expected	Observed
Runtime code	build of bfaaf70, no immutables	exact, 5680 bytes
Roles / storage	none	n/a

CCTP_FACET (`CCTPFacet.sol`):

Surface	Expected	Arbitrum	Base
Runtime code	build with verified messenger and USDC immutables	exact, 0xf552b3cf9556553ae22056ddb39a7e17a1ed9ad24c1396b032203b99c7e90099	exact, 0x64d4f4dbdf5fe32cb901e99131f71535c5460e5c3d9ef499cb665f7636a80571
`cctp()` (immutable)	Circle TokenMessengerV2 0x28b5a0e9C621a5BadaA536219b3a228C8168cf5d	match	match
`usdc()` (immutable)	native USDC	0xaf88d065e77c8cC2239327C5EDb3A432268e5831	0x833589fCD6eDb6E08f4c7C32D4f71b54bdA02913
`DESTINATION_CALLER()`	0	0	0
`MIN_FINALITY_THRESHOLD()`	2000	2000	2000
`VERSION()`	"1.0.0"	match	match

`toCCTPRateLimitKey()`	keccak256("LIMIT_USDC_TO_CCTP")	0x0476a9fd902eafdb5bcdabd9f0523dd7aacf7aa0c38c0e6ab912f5fed00f8e11	same
`getToDomainRateLimitKey(0)`	keccak256(abi.encode(keccak256("LIMIT_USDC_TO_DOMAIN"), 0))	0x90b598dae614822d554ec4f627bb5feab505106473a7911f34b36c0b3b9695c7	same

PAU_ACCESS_CONTROLS (`AccessControls.sol`):

Surface	Expected	Arbitrum	Base
Runtime code	build of cbf71b2, no immutables	exact, 1953 bytes	exact
`getRoleMemberCount(DEFAULT_ADMIN_ROLE)` / member 0	1 / SPARK_EXECUTOR	1 / 0x65d946e533748A998B1f0E430803e39A6388f7a1	1 / 0xF93B7122450A50AF3e5A76E1d546e95Ac1d0F579
`getRoleMemberCount(ALLOCATOR_ROLE)` / member 0	1 / PAU_ADMINISTERED_AGENT	1 / 0x0745aae633E8318a063D383791bCc0d8C82F46C6	1 / 0x70E46bAf2E3F27a119757D7b796c641C8bc087cE
`getRoleMemberCount(CONTROLLER)`	0 (role unused here)	0	0
`getRoleAdmin(ALLOCATOR_ROLE)`	DEFAULT_ADMIN_ROLE	0x00	0x00
deployer / factories	no role	false	false

PAU_RATELIMITS (`RateLimits.sol` , plain AccessControl, not enumerable):

Surface	Expected	Arbitrum	Base
Runtime code	build of cbf71b2, no immutables	exact, 2805 bytes	exact
`hasRole(DEFAULT_ADMIN_ROLE, SPARK_EXECUTOR)`	true	true	true
`hasRole(CONTROLLER, PAU_CONTROLLER)`	true	true	true
`hasRole(CONTROLLER, legacy controller)`	false	false	false
deployer / factory	no role	false	false
`getRateLimitData(LIMIT_USDC_TO_CCTP)`	all zero (pre-spell)	(0,0,0,0)	(0,0,0,0)
`getRateLimitData(domain-0 key)`	all zero	(0,0,0,0)	(0,0,0,0)

PAU_CONTROLLER (`Controller.sol`):

Surface	Expected	Arbitrum	Base

Runtime code	build with verified Beacon immutable	exact, 0x3e2532d63ed044c7342a37b4e538853c402ae4b886cab70a9181aa313d80ad2f	exact, 0x381238889ec756310e361b66c11bd0c8c67fe9893aabc812d9f3af3c63479078
`beacon()` (immutable)	SPARK_BEACON	match	match
`proxy()` (set once)	existing legacy ALM_PROXY	0x92afd6F2385a90e44da3a8B60fe36f6cBe1D8709	0x2917956eFF0B5eaF030abDB4EF4296DF775009cA
`accessControls()` (set once)	PAU_ACCESS_-CONTROLS	match	match
`rateLimits()` (set once)	PAU_RATELIMITS	match	match
`integrations().length` / id	1 / CCTP_FACET	1 / match	1 / match
`getConfig(CCTP_FACET)`	equals the Beacon's	facet and 10 wires identical	same
`getDispatch(callSelector)` for the 10 wrappers	(facet, delegateSelector)	match	match
`getDispatch(delegateSelector)` for the 10 raw selectors	unwired (0, 0)	(0, 0)	(0, 0)
`cctp_getDomainParameters(0)`	(0, 0, 0) pre-spell	(0,0,0)	(0,0,0)
Initializable / reentrancy guard	version 1, NOT_ENTERED	1 / 1	1 / 1

PAU_ADMINISTERED_AGENT (`AdministeredAgent.sol`):

Surface	Expected	Arbitrum	Base
Runtime code	build of bfaaf70, no immutables	exact, 5002 bytes	exact
`adminCount()` / `getAdmin(0)`	1 / SPARK_EXECUTOR	1 / 0x65d946e533748A998B1f0E430803e39A6388f7a1	1 / 0xF93B7122450A50AF3e5A76E1d546e95Ac1d0F579
`actorCount()` / `getActor(0)`	1 / ALM_RELAYER_-MULTISIG	1 / 0x8a25A24EDE9482C4Fc0738F99611BE58F1c839AB	same
`grantorCount()` / `getGrantor(0)`	1 / PAU_GRANTOR_-MULTISIG	1 / 0x4B61A0E48dd1e300f64090C60F414c1aC6CbC514	same
`revokerCount()` / `getRevoker(0)`	1 / ALM_FREEZER_-MULTISIG	1 / 0x90D8c80C028B4C09C0d8dcAab9bbB057F0513431	same
`getIsActor(backstop)`	false at the snapshots	false	false
Membership checks (admin/actor/grantor/revoker) for deployer, factories, zero address	all false	false	same

Other persistent config	none exist (no rate limit, target binding, allowlist, expiry, delay, pause)	none	same
Roles held elsewhere	ALLOCATOR_ROLE on PAU_ACCESS_-CONTROLS only	`hasRole` checks on the new contracts, not CONTROLLER on the proxy	same

Existing contracts (must be unchanged):

Contract	Surface	Expected	Arbitrum	Base
ALM_PROXY	`hasRole(DEFAULT_ADMIN_ROLE, SPARK_EXECUTOR)`	true	true	true
ALM_PROXY	`hasRole(CONTROLLER, legacy controller)`	true	true	true
ALM_PROXY	`hasRole(CONTROLLER, PAU_CONTROLLER)`	false until spell	false	false
ALM_PROXY	deployer roles	none	none	none
Legacy	`RELAYER(ALM_RELAYER_MULTISIG)`	true	true	true
Legacy	`RELAYER(ALM_BACKSTOP_RELAYER_MULTISIG)`	true	true	true
Legacy	`FREEZER(ALM_FREEZER_MULTISIG)`	true	true	true
Legacy	`mintRecipients(0)`	Ethereum ALM_PROXY	0x0000000000000000000000001601843c5e9bc251a3272907010afa41fa18347e	same
ALM_RATE_-LIMITS	`hasRole(CONTROLLER, PAU_CONTROLLER)`	false	false	false
ALM_RATE_-LIMITS	deployer transactions to it	none	none (nonce enumeration)	none
Legacy controller	`cctp()`	V1 messenger (unchanged)	0x19330d10d9cc8751218eaf51e8885d058642e08a	0x1682Ae6375C4E4A97e4B583BC394c861A46D8962
SPARK_-EXECUTOR	`delay()` / `gracePeriod()`	0 / 604800 s	0 / 604800 s	same

Deployer EOA	nonce after configure, no later tx	35 / 43	35	43

Multisigs referenced by the agent: Covered in the Safe table above (thresholds 1/2, 2/5, 2/4, 3/5. Identical owners and thresholds on both chains, no modules). Owner identities are not on chain.

Coverage and evidence limits: The default statements above rest on getter reads and configuration-event history at the locked blocks: zero rate-limit records on both PAU keys, zero domain-0 parameters on both controllers, and no rate-limit or domain-parameter configuration events since deployment. Enumerable role and membership counts are 1, and the listed negative role and membership checks confirm no residual deployer or factory authority. PAU_RATELIMITS role membership was reconstructed from events and checked with `hasRole`.

On Arbitrum the explorer's complete transaction list confirms the 23 deployment transactions and nothing else, none reverted. On Base the explorer's free tier does not serve transaction lists, so reverted third-party attempts there cannot be excluded from the collected evidence. They would have no effect. An exhaustive transaction-history claim is therefore not made.

Coverage	Arbitrum One	Base
New-contract deployment/setup transactions inspected	23 receipts and call traces	20 receipts and call traces, not an exhaustive transaction listing
Additional authority/configuration transactions inspected	79 receipts and call traces	320 receipts and call traces
Dependency state	166 getter checks across 16 dependencies	166 getter checks across 16 dependencies

Planned activation: The planned activation moves SPARK_BEACON admin from SPARK_EXECUTOR to each chain's `SKY_GOV_RELAY`: Arbitrum in the October 8 spell, Base in its own activation spell. Spark Executor keeps the admin of PAU_ACCESS_CONTROLS, PAU_RATELIMITS and the agent contract on both chains. A new facet would require Sky to register it in the Beacon and a PAU administrator to sync it into the Controller. This changes future facet governance without changing the installed CCTP configuration.

For Arbitrum, the planned actors are ALM_RELAYER_MULTISIG and the Planner's single-key hot wallet, which calls the agent directly. The backstop is not added. A Soter grantor replaces PAU_GRANTOR_MULTISIG, a Soter revoker is added alongside the existing revoker, and the PAS Configurator receives DEFAULT_ADMIN_ROLE on both PAU_ACCESS_CONTROLS and PAU_RATELIMITS. A grantor can add its own Safe as an actor, and Spark's agent admin can also add actors.

The October 8 draft at `0cb7b26` contains this Arbitrum configuration, but the relay, PAS and new operator addresses are TODO placeholders. The agent has no selector allowlist; the reviewed Controller permissions and CCTP configuration bound its actors' access to proxy funds. A Base activation payload was not located in the public spell material checked.

Activation materially changes the deployed permission and recovery model and requires verification, including the PAS instance's permitted calls and rate-limit policy. This attestation's verdict covers the configuration snapshot blocks. Targeted reads at Arbitrum block `509616865` and Base block `51892763` showed the Agent memberships listed above, executor-only Beacon and AccessControls admin, no PAU proxy authority, zero domain-0 parameters and zero records for both PAU CCTP limit keys. All sixteen scoped runtimes were unchanged. These targeted reads do not cover the full configuration history or provide approval of either activation payload.

Not covered here: The PR's Immunefi-listing checklist item cannot be checked on chain and was unticked when the PR was reviewed.

Approval of the activation spell (CONTROLLER grant on the proxy, domain-0 recipient and fee caps, PAU limits, Agent memberships, Beacon handover and PAS grants), the resulting combined legacy-plus-PAU

capacity, Safe owner identities, exhaustive dependency transaction history, and Circle/USDC contract internals and upgrade paths.

After the spell, re-check:

1. Both CONTROLLER memberships on each proxy.
2. Still exactly one integration and ten wires on each controller. Facet and all runtimes unchanged, immutables still the intended addresses.
3. Domain-0 recipient is the legacy mainnet `ALM_PROXY` `0x1601843c5E9bC251A3272907010AFa41Fa18347E`, fee caps as intended, and no other domain configured on either controller.
4. Both PAU limit records (max, slope, lastAmount, lastUpdated) and the legacy records, then check combined capacity and refill rates against the system-wide four-route policy.
5. Agent actors, grantors and revokers match the final approved addresses and counts. Spark retains agent admin. Verify the Planner account and the new Soter Safes, including their thresholds, owners, guards and modules, separately from the old Safe checks. Confirm no unapproved actor was added.
6. Beacon admin is the intended Sky relay and Spark's Beacon admin is removed. Spark retains admin of AccessControls and RateLimits; verify the PAS grants on both, the deployed Configurator and its BeamState, authorized callers, permitted Controller calldata and rate-limit policy.
7. Proxy USDC allowances to both CCTP messengers are zero; any pending CCTP messages are accounted for.
8. Circle dependency state unchanged: implementations, thresholds, per-message caps, denylist.

4 Parallel Controller Architecture Validation

4.1 Summary

Spark has put a second controller next to the legacy `ForeignController` on Arbitrum One and on Base. On each chain, both controllers are bound to the same existing `ALMProxy`, which holds the funds. The new controller is a Diamond PAU `Controller` with one wired facet, `CCTPFacet`, for initiating USDC transfers through Circle's CCTP V2 to Spark's Ethereum `ALMProxy`. At the reviewed configuration blocks (Arbitrum One `507725257`; Base `51638244`), the new PAU route was inactive: its controller had no proxy authority, its destination was unset and its rate limits were zero. The activation spell had not run at those blocks.

We verified the deployed state on both chains and reviewed how the two controllers would interact after the planned activation. We approve the deployments in their reviewed pre-spell state and the CCTP-only parallel architecture under the conditions below.

Contracts and token, bridge and investment integrations are assumed to behave as specified. Contract-level defects are outside scope. Governance and administrators are trusted. The review covers configuration consistency, operator mistakes, misuse of granted permissions, incomplete shutdowns and inconsistent budgets.

4.2 Why a second controller

The deployed legacy controllers are bound to Circle's CCTP V1. Their messenger addresses are immutable and their calls use the V1 signature, so V2 cannot be added to those deployments by configuration. Spark's planned V2 route uses the diamond-pau `CCTPFacet`. On each L2, either route spends USDC from the existing `ALMProxy`, so Spark bound the new controller to that proxy instead of deploying a fresh one. The planned destination mint is to Spark's separate Ethereum `ALMProxy`.

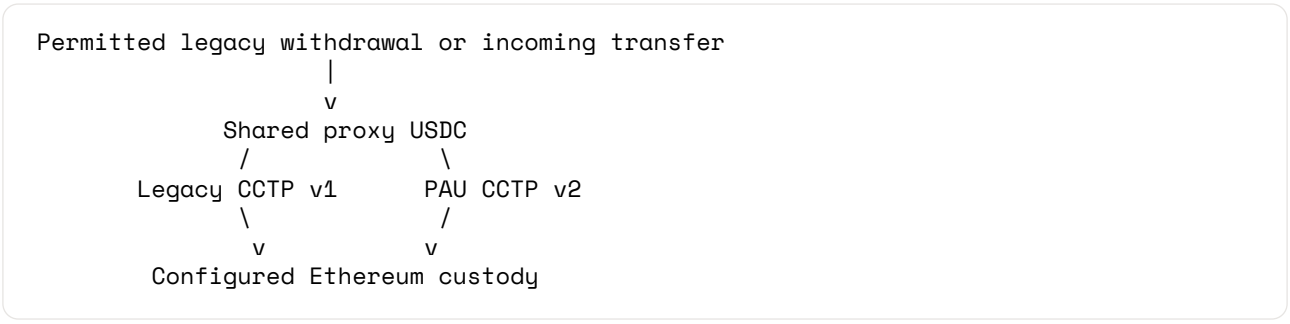
4.3 How the two controllers work

After activation, each controller authorizes its own operations against the same funds, without the other's approval. `CONTROLLER` on `ALMProxy` is unrestricted: any holder may call `doCall`, `doCall`

`WithValue` and `doDelegateCall` with arbitrary target and data. What limits the new controller is its own dispatch table and role checks. Its fallback routes the ten wired `cctp_` selectors synced from the Beacon and rejects other unwired selectors; the controller's direct admin and getter functions remain callable separately. Of the ten wired selectors, two change state: `cctp_setDomainParameters`, admin only, and `cctp_transfer`, allocator only. The legacy controller retained its existing functions and permissions at the reviewed blocks.

The same operator keys are configured for both. `ALM_RELAYER_MULTISIG` holds `RELAYER` on the legacy controller and is the actor on the agent that holds the PAU allocator role. The PR plans to add `ALM_BACKSTOP_RELAYER_MULTISIG` as a second actor. Once the PAU route is activated, anyone who can act through an authorized Safe can use either route. No second approval sits between the two routes. The Safe's own signer threshold still applies.

What the PAU route can and cannot do after activation. It can spend the proxy's native USDC through its fixed Circle V2 messenger, to an administrator-configured recipient, within its own total and per-destination limits. The operator cannot pick a recipient through `cctp_transfer`, spend another asset through the wired facet, or redeem legacy positions. Under the PR's plan both routes send to Spark's Ethereum `ALMProxy` `0x1601843c5E9bC251A3272907010AFa41Fa18347E`, with the PAU fee caps at zero.



Who controls what, on each chain. Roles and limits are the verified pre-spell state; PAU bridging in the table assumes the planned activation.

Party	Legacy side	PAU side
`SPARK_EXECUTOR` (governance relay executor, delay 0 at the reviewed blocks)	admin of the controller, the proxy and the rate limits	sole admin of Beacon, AccessControls, RateLimits and the agent at the reviewed blocks
`ALM_RELAYER_MULTISIG` (Safe, 1 of 2)	`RELAYER`, may bridge	actor on the agent, which is the only allocator; may bridge after activation
`ALM_BACKSTOP_RELAYER_MULTISIG` (Safe, 2 of 5)	`RELAYER`, may bridge	to be added as actor by the spell
`ALM_FREEZER_MULTISIG` (Safe, 2 of 4)	`FREEZER`, may remove relayers	revoker on the agent, may remove actors
`PAU_GRANTOR_MULTISIG` (Safe, 3 of 5)	none	grantor on the agent, may add actors
Rate limits	legacy `RateLimits`, towards Ethereum 50m cap on Arbitrum, 200m on Base, aggregate unlimited at the reviewed blocks	own `RateLimits`, all keys unset at the reviewed blocks

On each chain, the executor is the administrator of both controller systems at the reviewed blocks. Its delay was zero and no guardian was configured in the reviewed state; the governance delay is upstream on mainnet. The planned PAS Configurator grant would add another PAU administrator and is outside the verified pre-spell state.

4.4 Risks

The proxy has no per-Controller asset partition, reserve floor or combined rate limit; those controls have to come from the Controllers and their configuration.

The cases below concern the planned activated CCTP-only setup. At the reviewed blocks, the PAU route could not transfer funds.

Independent budgets apply to the same funds:

- Each route has a total CCTP limit and a destination limit, in separate RateLimits storage. Per destination, before refill, each route's capacity is the smaller of its two remaining limits. The two capacities add; actual outflow is bounded by available USDC and successful bridge execution.
- V1 spending leaves V2 capacity unchanged. A single Controller could already exhaust liquid USDC if its limit permitted; the second adds capacity against the same balance. Duplicating the full budget across both routes doubles configured capacity when their capacities match.
- Split the shared-proxy budget between the routes to keep their combined capacity within the approved combined budget. Approve their limits and refill rates together, including other enabled outflows.
- Any required USDC reserve for existing operations has to be enforced across all enabled outflows. Neither CCTP route checks a minimum remaining balance.

Example numbers, not deployed limits. Amounts are whole USDC, both routes are enabled for one destination, and limits do not refill during the check. Each `min` takes the smaller of a route's remaining total and destination capacity.

```
v1_remaining = min(6_000_000, 4_000_000)  # 4,000,000 USDC
v2_remaining = min(4_000_000, 3_000_000)  # 3,000,000 USDC
combined_capacity = v1_remaining + v2_remaining
approved_combined_capacity = 5_000_000
within_budget = combined_capacity <= approved_combined_capacity  # False
```

Legacy withdrawals also fund the new route:

- Legacy PSM withdrawals return USDC to the shared proxy, where either bridge route can spend it regardless of its intended local use, such as a deposit or repayment. Withdrawal followed by V1 bridging already existed; PAU adds an independently budgeted outlet for the same funds.
- The PSM withdrawal limit bounds the funding operation only; it does not consume PAU's bridge budget or reserve the proceeds.
- PSM shares, permitted investment exits and incoming transfers can fund a proxy that has zero liquid USDC. Available amounts and timing bound outflows; recovered funds remain exposed to every enabled outflow until spent or protected by an enforceable restriction.
- PAU cannot force those legacy withdrawals, but an operator authorized on both systems can run both halves of the workflow.

Illustrative reserve check for a proposed outflow, assuming no other balance changes. This is a policy check; neither CCTP function currently enforces it.

```
liquid_usdc = 8_000_000
required_reserve = 2_000_000
proposed_outflow = 7_000_000
reserve_preserved = liquid_usdc - proposed_outflow >= required_reserve  # False
```

A one-route shutdown leaves the other route usable:

- Removing an identity from the Agent stops its PAU operations but leaves its legacy relayer permission unchanged, and vice versa.
- Disabling one CCTP route does not disable the other. Incoming or recovered USDC stays available to the route that remains authorized.
- Incident response has to distinguish removing one operator from a complete halt:

  - A complete operator halt covers primary and backstop identities on both authorization surfaces on both chains.
  - A bridge-specific halt covers both CCTP versions while keeping any required non-bridge exits open.
- Changing the Beacon alone does not remove dispatch already installed on a Controller.
- During a V2 outage, failover onto unused V1 capacity has to stay inside the combined spending decision and account for transfers already submitted.
- Both versions still draw on the same source USDC balance, Circle settlement and destination custody.

Separate source transactions can fund the same intended transfer twice:

- The Controllers share no business-transfer identifier. A successful source burn stays committed while the destination mint is pending.
- Treating that delay as a failure and resubmitting through the other route can create a second transfer if the balance and limits permit it. Both local limit checks can pass.
- Duplicate submission is possible on one route too. The second route adds separate permissions, capacity and transaction records that have to be reconciled against the same business instruction. These are two distinct source burns, not a replay of one Circle message.
- Before retrying, operations have to reconcile source acceptance and destination settlement across both versions, including every message of a chunked transfer.
- Limits refill with time, not on destination settlement, so outstanding transfers can accumulate across refill periods. Shutting down the source does not cancel burns already accepted.
- A failed PAU transfer reverts its own approvals, burn and rate-limit changes; it does not undo an earlier successful legacy withdrawal or bridge transaction. Recovery has to start from the committed state of each transaction.

A delegated EOA facet can change its own code after admission:

- Facet admission in the Beacon and the Controller's Beacon sync accept any address whose runtime code is non-empty. On a chain where EIP-7702 delegation is active, a delegated EOA carries a 23-byte delegation marker as its code and passes that check.
- The EOA's signer can repoint the delegation at any time without the Beacon admin or the Controller admin. The fallback delegatecalls the stored facet address with no further code check, and role checks live in facet code, so any caller of a wired selector runs the replacement code with Controller storage access. It can rewrite the dispatch table, the proxy address and the rate limits.
- Only an authorized admin admitting a delegated EOA as a facet opens this path, which the reviewed configuration has not done. Likelihood is low; the effect is full control of that Controller.
- Admission and sync should reject delegation-marker code (the `0xef0100` prefix); if facet code identity must stay fixed, compare an expected code hash at dispatch. A code-existence check at dispatch would also stop a facet whose code was cleared from reporting silent success.

4.5 Adding or retiring functionality

Adding a function: Every new function on either controller is another way to spend the same funds. A separate controller, facet or rate-limit key does not create separate assets, it only adds a new way of using the funds.

Before a new function goes live, check what it can touch and compare it with every existing route over the same assets:

- Which assets it can move.
- Which destinations it can send to.
- Which external contracts it approves or calls.

- How it keeps its books.

Then set **one combined budget** for old and new together, and make sure enough assets stays on the chain for local operations and for exits. Check that nothing the new function may do can break an assumption the other controller relies on, such as a balance it expects to hold or a position it expects to still have.

Repeat this review whenever a facet or a material permission changes. **The approval given here covers the CCTP facet alone.**

Moving a function from the legacy controller to PAU: While both versions of a function are live, treat them as one budget. Spending that already happened and transfers still in flight count against it, a new route that starts with a full limit ignores that history. The same applies when a route that was switched off is switched back on, its stored limit may have refilled in the meantime.

Keep the legacy controller's authority for as long as it is needed to finish existing positions and obligations. Switch off new activity in a way that leaves those exits working.

> **One concrete trap in the deployed legacy code.** When Aave or ERC-4626 positions are withdrawn, the controller credits the matching deposit limit. If that deposit limit was set to zero to stop new deposits, the withdrawal reverts even though the withdrawal limit is open.

Before retiring the function:

1. Account for its outstanding work.
2. Remove authority and allowances that are no longer needed.
3. Confirm that the replacement does not leave a second, unintended way to spend.

A rollback needs the same budget and pending-work checks.

Retiring the legacy Controller: Take away the legacy controller's access to the proxy only once everything it still does has been moved, finished or given another working path. The new controller only bridges USDC. It cannot manage or unwind other legacy positions.

Both controllers use the same proxy, so switching authority does not move any assets. Revoking the role stops the legacy controller from making new proxy calls. Bridge messages accepted before revocation remain eligible for destination settlement, subject to Circle and destination conditions, and existing obligations still stand.

Token allowances and permissions that other contracts hold are not affected by the revocation. Remove them in their own step, and keep the ones still needed to finish outstanding work.

4.6 Assessment

On both chains, the planned CCTP-only setup does not give an operator a new destination they control for stealing funds, under the stated assumptions. The operator chooses an enabled domain, amount and fee cap within the configured range; PAU administrators configure the recipient on the destination chain.

After activation, an operator can send more USDC to Ethereum than the approved liquidity policy allows, or send it at the wrong time. With the planned recipient configured, the funds are destined for Spark-controlled custody, but the transfer can still cause material L2 liquidity disruption. Once the bridge transfer completes, the USDC is in Spark's mainnet proxy. Getting it back to the L2 requires another authorized transfer and another wait for the bridge to complete.

The budget, shutdown and retry points above are rules for how the routes are operated, not bugs. Before activation, check that the spell's settings keep spending within the budget governance approved. At the reviewed blocks, the new controller had no proxy authority or configured limits. The production limits and required L2 USDC reserves remain to be decided and verified against the activation payload.

4.7 Final statement

Cantina has reviewed the Arbitrum One and Base parallel controller deployments in sparkdotfi/spark-address-registryPR 115, including the parallel controller architecture from first principles.

- **Arbitrum One deployment: approved in its reviewed pre-spell state.**
- **Base deployment: approved in its reviewed pre-spell state.**
- **Parallel controller architecture on both chains: approved for the CCTP-only configuration described in this report, with the conditions below.**

Before the activation spell runs, check four things:

1. **The combined budget is written down.** Legacy and PAU limits and refill rates, added together with every other outflow, stay within the liquidity policy governance approved. When capacity moves from legacy to PAU, past spending and transfers in flight count against it. Moving to PAU should not give the operator a fresh spending budget on top of what was already used through the legacy route.
2. **The spell sets the approved values.** Check the executed state against the decoded payload and approved recipient, fee caps, enabled domains and actor memberships. The PR description alone does not prove execution. If the PAS Configurator is granted a role, check what that role can actually call.
3. **Incident response covers both routes.** There is one procedure to remove an operator and another to stop bridging. Both name the primary and the backstop, both chains, and who can restore access afterwards.
4. **Retries look at both CCTP versions.** Before a transfer is resubmitted or moved to the other route, operations confirm what was burned at the source, what settled at the destination, and every message of a chunked transfer.

These checks run against the activation payload and the state it produces. This report does not approve that payload. Adding another facet or changing a material permission requires a new assessment.

Rollout recommendation. Start the PAU route with a small limit inside the combined budget while V1 carries the volume. Once it has run without issues, move capacity over. If V1 is being retired, set its CCTP limit to zero and keep only the legacy functions still needed to unwind positions.

4.8 Exception

Bugs not yet found in either controller, the shared `ALMProxy` or their dependencies could still put the funds at risk, even with correct permissions and configuration.