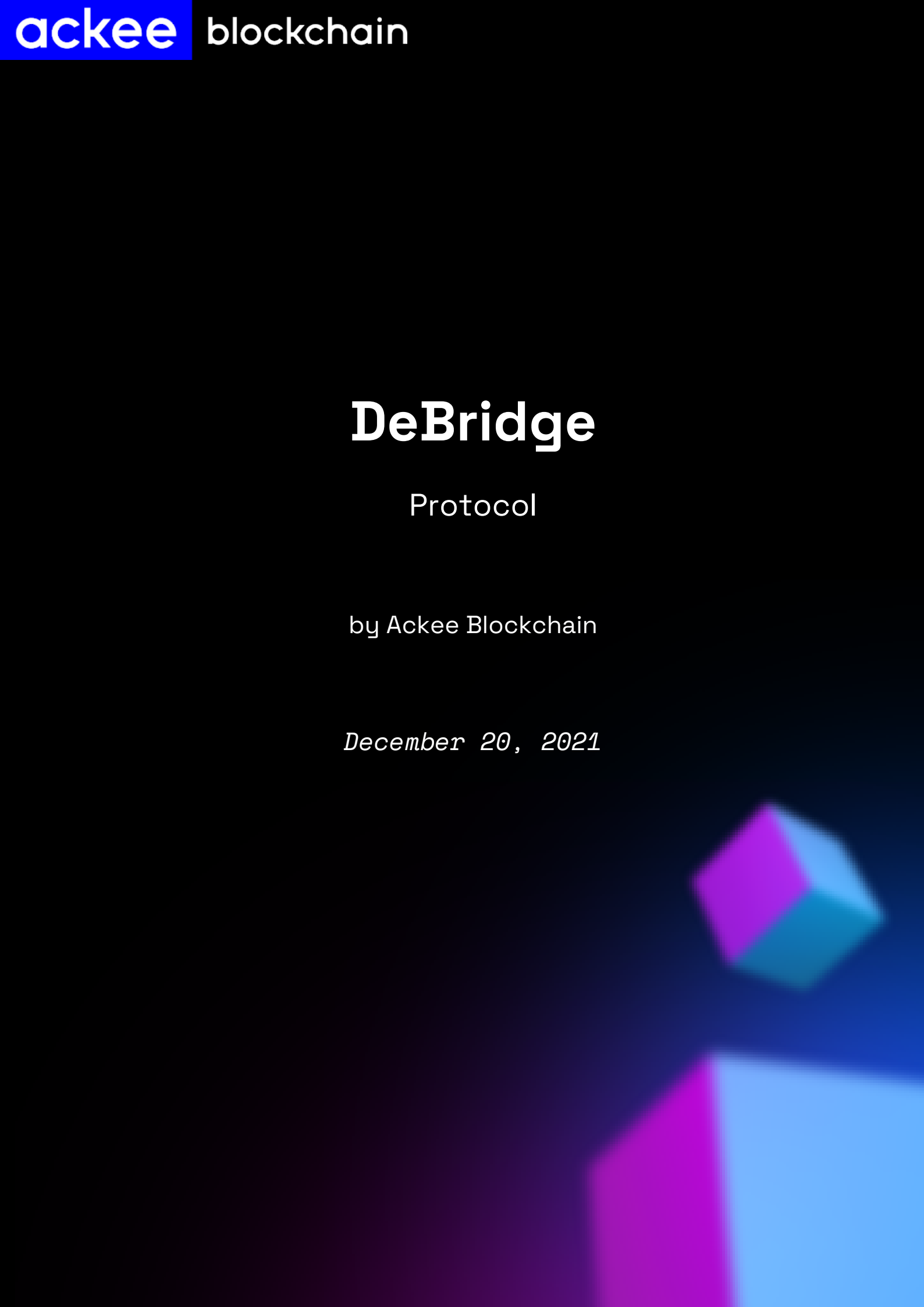
ackee blockchain

DeBridge

Protocol

by Ackee Blockchain

December 20, 2021



Contents

1. Document Revisions

1.0	Revision 1 - initial document	Oct 1, 2021
2.0	Revision 2 - re-audit	Dec 6, 2021
2.1	Final report of Revision 2	Dec 20, 2021
2.2	Complementary audit - Crates	Sep 22, 2022
2.3	Fix review	March 17, 2023

2. Overview

This document presents our findings in reviewed contracts.

2.1. Ackee Blockchain

Ackee Blockchain is an auditing company based in Prague, Czech Republic, specializing in audits and security assessments. Our mission is to build a stronger blockchain community by sharing knowledge – we run free certification courses School School of Solana, Summer School of Solidity and teach at the Czech Technical University in Prague. Ackee Blockchain is backed by the largest VC fund focused on blockchain and DeFi in Europe, Rockaway Blockchain Fund.

2.2. Audit Methodology

The Ackee Blockchain auditing process follows a routine series of steps:

1. **Code review**
    a. High-level review of the specifications, sources, and instructions provided to us to make sure we understand the project's size, scope, and functionality.
    b. Detailed manual code review, which is the process of reading the source code line-by-line to identify potential vulnerabilities. We focus mainly on common classes of Solana program vulnerabilities, such as:

    missing ownership checks, missing signer authorization, signed CPI of unverified programs, cosplay of Solana accounts, missing rent exemption assertion, bump seed canonicalization, incorrect accounts closing, casting truncation, numerical precision errors, arithmetic overflows or underflows, ...

c. Comparison of the code and given specifications, ensuring that the program logic correctly implements everything intended.

d. Review of best practices to improve efficiency, clarity, and maintainability.

2. **Testing and automated analysis**

   a. Run client's tests to ensure that the system works as expected, potentially write missing unit or fuzzy tests using our testing framework Trdelnik.

3. **Local deployment + hacking**

   a. The programs are deployed locally, and we try to attack the system and break it. There is no specific strategy here, and each project's attack attempts are characteristic of each program audited. However, when trying to attack, we rely on the information gained from previous steps and our rich experience.

2.3. Review team

Member's Name	Position
Tibor Tribus	Lead Auditor
Vladimír Marcin	Auditor
Adam Hrazdira	Auditor
Josef Gattermayer, Ph.D.	Audit Supervisor

2.4. Disclaimer

We've put our best effort to find all vulnerabilities in the system, however our findings shouldn't be considered as a complete list of all existing issues. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

3. Executive Summary

This document presents a security audit of the DeBridge protocol. An initial audit (see Revision 1) has been performed between October 18, 2021 and October 29, 2021. The time donation was 20 engineering days. Several issues were identified and reported to the client. This audit revision was agreed as an internal audit, therefore we don't comment code maturity or test coverage of this version in the report.

Update Nov 22, 2021: The re-audit (see Revision 2) took place between November 22, 2021 and December 3, 2021 with a time donation of 10 engineering days. The main focus of re-audit was to check whenever reported issues from the Revision 1 were correctly fixed. In the remaining time we focused on two main objectives. First one was a re-audit of code changes caused by refactoring of existing functionality as it could also bring unintended bugs and security issues. The second objective was an audit of newly added functionality.

DeBridge correctly fixed all issues discovered in the Revision 1 except the issue M1, which was addressed correctly but not all of its occurrences. Revision 2 of the audit discovered three low and one high severity issues.

Update Sep 21, 2022: DeBridge provided an updated codebase, and we have performed a complementary audit of crates modules (see Complementary audit). The security review took 12 engineering days.

We also provided a quick code review of the rest of the repository that was outside this audit revision's scope. The performed review is not as detailed as an audit due to multiple changes in the code base since the last full audit. That is, the changes in the rest of the repository should not be considered

fully audited in this report.

Our security review resulted in 1 informational finding and 1 warning.

Update March 17, 2023: DeBridge provided an updated codebase and we have performed a fix review of previously reported issues (see Revision 2.3). All impactful issues were correctly fixed however the codebase changed significantly compared to the previous revisions and therefore an introduction of new vulnerabilities cannot be guaranteed as validation of the new code was not a scope of this fix review.

We recommend a new re-audit of the latest codebase.

4. Scope

This chapter describes the audit scope, contains provided specifications, supporting documentation and sets the main objectives for the audit process.

4.1. Coverage

Oct 1, 2021

Sources revision used during Revision 1 process:

- Repository: https://github.com/debridge-finance/solana-contracts
- Commit: `d86311af7b5c46e6324de5c912a6adcf35d35dd0`

Update Nov 22, 2021

After an agreement with the client we did not audit any external call logic. If any of this logic was used in the audit scope we have just assumed it's safe.

The following files were audited:

- `programs/settings/*`
- `programs/debridge/*`

The following code was excluded from the scope of the audit:

- `programs/debridge/lib.rs`
  - `pub fn init_external_call_storage(…)` (1)
  - `pub fn update_external_call_storage(…)` (1)
  - `pub fn close_external_call_storage(…)` (1)

- `pub fn execute_external_call(…)` (1)
- `mod external_call_storage{…}`

Sources revision used during the Revision 2 process:

- Repository: https://github.com/debridge-finance/solana-contracts
- Commit: `94b840c5742086de20eaa741eb8958b31574bb1f`

(1) We've also excluded the contexts used by the instructions.

Update Sep 21, 2022

Sources revision used during Complementary audit process:

- Repository: https://github.com/debridge-finance/solana-contracts
- Commit: `e262a1140cd27098c3e819095def52b78d40a3c0`

Files being audited :

- `solana-contracts/crates/crypto/src/u256.rs`,
- `solana-contracts/crates/debridge-external-call/src/cost_calculation.rs`,
- `solana-contracts/crates/debridge-external-call/src/external_instruction.rs`,
- `solana-contracts/crates/signature-verifier/lib.rs`,
- `solana-contracts/crates/submission/src/lib.rs`,

Update March 17, 2023

Sources revision used during Revision 2.3 process:

- Repository: https://github.com/debridge-finance/solana-contracts
- Commit: `22bc8b2a3556cd60d1f965fe674183e22a05aa41`

4.2. Supporting Documentation

Most of the documentation is located in the Gitbook of the project. It was helpful to understand the overall architecture design.

Another documentation is available in the form of in-code comments (`RustDocs`) for `/program/settings/lib.rs` and `/program/debridge/lib.rs`. Readme of the project is extensive and covers the basic workflow of the protocol.

4.3. Objectives

We've defined the following main objectives of the audit:

- Check the overall code quality and best practices.
- Check functionality of the system.
- Check if nobody unauthorized is able to claim or send assets.
- Check if the discovered issues were correctly fixed and check the correctness of a newly implemented logic for the change balance and keys management.
- Check if the refactored code didn't bring any new issues.

5. System Overview

This section contains an outline of the audited contracts. Note that this is meant for understandability purposes and does not replace project documentation.

The basic idea behind the DeBridge protocol is to allow a transfer of assets between various blockchains. It is achieved by locking/unlocking an asset on a native chain and minting/burning the wrapped asset on a secondary chain. From a smart contracts perspective, Solana is acting once as a native chain and once as a secondary chain, from this point of view, there are four possible scenarios:

1. Transfer Solana SPL token from Solana to a secondary chain.
2. Transfer Solana SPL token back to Solana chain from a secondary chain.
3. Release asset on a native chain, and burn deAsset on Solana chain.
4. Transfer asset from a native chain, and mint deAsset on Solana chain.

The verified system behaves differently in each of these scenarios. The behavior depends on a type of a bridge and a type of an invoked method. We will now discuss these scenarios in more detail.

`SendBridge` ≡ SOLANA is NATIVE:

1. **Locking (staking)** (1)
    - A transferred amount of an asset is locked by sending it to a wallet with which can operate only a smart contract.
    - *Mint deAssets on a secondary chain.* (2)
2. **Releasing** (3)
    - *Burn deAsset on a secondary chain.* (2)

- Transfer Solana SPL tokens from a wallet owned by a smart contract to a user wallet at the Solana chain.

`MintBridge` ≡ SOLANA is SECONDARY:

3. **Burning** (1)
   - A particular amount of deAsset is burned from a sender wallet on the Solana chain.
   - *Release native assets on a native chain.* (2)
4. **Minting** (3)
   - *Lock native assets on a native chain.* (2)
   - A transferred amount of deAsset is minted and sent to a wallet owned by a user.

---

(1) Invoking the `send` instruction.

(2) This is happening on other chains (not `Solana`).

(3) Invoking the `claim` instruction.

5.1. Key Components

The project is divided into two programs: `settings` and `debridge`. Settings program is responsible for initialization and managing a state of the protocol. The DeBridge program is responsible for asset flow between chains.

We have identified the following key components in the system. And that's what we tried to focus on the most.

Bridge is represented by `Bridge` structure in `settings/src/bridge.rs [loc:`

`225]`

- Holds information about a type of a bridge (`sendBridge`, `mintBridge`), and a state of the bridge.

State is represented by `State` structure in `settings/src/state.rs [loc: 368]`

- Holds all information about the state of the system. Defines who is `Authority`, `StopTap`, `FeeBeneficiar`. Store oracles addresses and information necessary to validate transactions.

Send instruction in `debridge/src/lib.rs [loc: 208]`

- Is invoked by a user when he wants to initialize a transfer of assets from the Solana chain. This instruction can operate in two different modes (locking assets/burning assets) based on a bridge type.

Claim instruction in `debridge/src/lib.rs [loc: 351]`

- Is invoked by a user when he wants to transfer assets to the Solana chain. This instruction can operate in two different modes (releasing assets/minting assets) based on a bridge type.

An overview of all components can be found in diagrams in Appendix G and Appendix G.

5.2. Actors

This part describes actors of the system, their roles, and permissions.

Protocol Authority

Protocol authority is represented by an account that is associated with a state of the system. It's responsible for initiating and managing the state and also for fees collection. It's the account with the highest permission level in

the system.

Protocol

The protocol itself is the owner of a wallet in which the protocol holds a locked asset and is, therefore, the only one who can release the assets. It is also responsible for minting new tokens and only the protocol can give consent for minting.

Stop Tap

The account has the privileges to stop the protocol but does not have the authority to start it. This account can be changed by the `Protocol Authority`.

User

A user of DeBridge protocol. This account can initialize Bridge, and invoke `Send`, `Claim` instructions, however it can manipulate only assets owned by him.

Claimer

This role is used for automatic claims on behalf of the user of DeBridge protocol. But just in case the user pays execution fees in advance.

5.3. Trust model

Users have to trust developers that logic and math algorithms are correctly implemented and they receive the correct amount of Assets or deAssets.

The user must also trust the validators not to team up and thus attack the protocol. However, the DeBridge protocol implements delegated staking and slashing mechanics that act as a backbone for the protocol security and prevent economic incentives for validators to collude.

6. Vulnerabilities risk methodology

A *Severity* rating of each finding is determined as a synthesis of two sub-ratings: *Impact* and *Likelihood*. It ranges from *Informational* to *Critical*.

If we have found a scenario in which an issue is exploitable, it will be assigned an impact rating of *High*, *Medium*, or *Low*, based on the direness of the consequences it has on the system. If we haven't found a way, or the issue is only exploitable given a change in configuration (such as deployment scripts, compiler configuration, use of multi-signature wallets for owners, etc.) or given a change in the codebase, then it will be assigned an impact rating of *Warning* or *Info*.

Low to *High* impact issues also have a *Likelihood*, which measures the probability of exploitability during runtime.

6.1. Finding classification

The full definitions are as follows:

Severity

		Likelihood			
		High	**Medium**	**Low**	**-**
Impact	**High**	Critical	High	Medium	-
	Medium	High	Medium	Medium	-
	Low	Medium	Medium	Low	-
	Warning	-	-	-	Warning
	Info	-	-	-	Info

Table 1. Severity of findings

Impact

- **High** - Code that activates the issue will lead to undefined or catastrophic consequences for the system.
- **Medium** - Code that activates the issue will result in consequences of serious substance.
- **Low** - Code that activates the issue will have outcomes on the system that are either recoverable or don't jeopardize its regular functioning.
- **Warning** - The issue cannot be exploited given the current code and/or configuration (such as deployment scripts, compiler configuration, use of multi-signature wallets for owners, etc.), but could be a security vulnerability if these were to change slightly. If we haven't found a way to exploit the issue given the time constraints, it might be marked as a "Warning" or higher, based on our best estimate of whether it is currently exploitable.
- **Info** - The issue is on the borderline between code quality and security. Examples include insufficient logging for critical operations. Another example is that the issue would be security-related if code or configuration (see above) was to change.

Likelihood

- **High** - The issue is exploitable by virtually anyone under virtually any circumstance.
- **Medium** - Exploiting the issue currently requires non-trivial preconditions.
- **Low** - Exploiting the issue requires strict preconditions.

7. Findings

This section contains the list of discovered findings. Unless overriden for purposes of readability, each finding contains:

- a *Description*,
- an *Exploit scenario*, and
- a *Recommendation*

Many times, there might be multiple ways to solve or alleviate the issue, with varying requirements in terms of the necessary changes to the codebase. In that case, we will try to enumerate them all, making clear which solve the underlying issue better (albeit possibly only with architectural changes) than others.

7.1. General Comments

This section presents an overall engineering culture that is a crucial precursor of the right security.

Overall code quality

An overall code quality is solid, the code is well written and understandable. We haven't found any major bad practices.

Commit culture

Developers successfully adopted a standardized commit message format that contains a name of a component and a short description.

Individual commits are small with descriptive commits titles which makes understanding of the repository much easier.

Some examples:

- `commit c2777ebef67a43bf66c25e4fe0b1613f286aa9aa`
  - 2 changed files, 3 additions, 3 deletions
  - Clearly described commit msg
- `commit d02d80bb6b7ba091eba00fdffb304adb7b49344b`
  - 2 changed files, 1 additions, 4 deletions
  - Clearly described commit msg

Comments and documentation

There is sufficient documentation describing how the program is supposed to work in general. Since the initial audit the Gitbook documentation was updated and extended, but we recommend adding more description of different fees that exist in the protocol and from what assets they are deducted. There should be a more detailed description of how validators are selected.

We have found several places in the code where comments didn't match the code as well, but it wasn't crucial for the audit process. We believe those mistakes were introduced during the refactor process.

Release cycle

The project contains a clear roadmap of stable releases. The client has started using the Github project where all the necessary information about the state of the current release cycle can be found. The client has already made the first stable release.

Code maturity

The code maturity got improved between two audit revisions. The client delivered a commit for the second revision of the audit on time and also added better documentation and extensive test coverage. The code was

functional with all tests successfully passing.

7.2. Revision 1 - Summary of Findings

	Severity	Impact	Likelihood
H1: `InitSendBridge` computational budget	High	Medium	High
H2: Custom program errors in `settings` program	High	Medium	High
H3: The program violates the stack size at runtime	High	Medium	High
M1: Use `create_program_address` instead of `find_program_address`	Medium	Medium	Medium
M2: Using API calls instead of `SysVar`	Medium	Medium	Medium
M3: Extra SEED during checking	Medium	Medium	Medium
M4: Comparing bad PubKeys	Medium	Medium	Medium
M5: Badly calculated rent exempt for one day	Medium	Medium	Medium
M6: `BridgeCtx::staking_wallet` bad constraint	Medium	Medium	Medium
L1: Typos in several places in code	Low	Low	Low

	Severity	Impact	Likelihood
L2: Bad naming conventions	Low	Low	Low
L3: NewTypes or type aliases for primitive types	Low	Low	Low
L4: Missing or Unused code	Low	Low	Low
L5: Unused accounts	Low	Low	Low
L6: Unconstrained `authority`	Low	Low	Low
L7: Using of `ProgramAccount` struct	Low	Low	Low
L8: Add extra optimizations in `Cargo.toml`	Low	Low	Low
L9: Use the latest stable Rust version (1.56)	Low	Low	Low
L10: Consider more pedantic `clippy` rules	Low	Low	Low

Table 2. Table of Findings

H1: `InitSendBridge` computational budget

High severity issue

Impact:	Medium	Likelihood:	High
Target:	settings/src/lib.rs, L#1304	Type:	Compute budget

Description

The `initialize_send_bridge` instruction consumed all computational units. The problem occurred during CPI from the `init_token_staking_wallet()` function. The issue can be fixed by creating a `staking_wallet` off-chain. You can find a test instruction in Appendix C.

```
"Program BkE5dgkHHhQZZCEjqDsRSLrkfZGYbeSWAjBzRSd2vyzh consumed 200000 of
200000 compute units"
```

Go back to Findings Summary

H2: Custom program errors in `settings` program

High severity issue

Impact:	Medium	Likelihood:	High
Target:	settings/src/lib.rs, L#701, 702, 775, 899	Type:	Account constraint

Description

These errors were caused by allocating insufficient space for accounts being created. They can be fixed by increasing the allocated space (`mem::size_of`). Also, one of these errors was caused by the use of a bad constraint. You can find a test instruction in Appendix D.

```
constraint = bridge_fee.to_account_info().is_not_exists()
```

Go back to Findings Summary

H3: The program violates the stack size at runtime

High severity issue

Impact:	Medium	Likelihood:	High
Target:	debridge/src/lib.rs, L#145	Type:	Stack size violation

Description

This error is caused by calling the `send()` instruction. You can find a test instruction in Appendix E.

Go back to Findings Summary

M1: Use `create_program_address` instead of `find_program_address`

Medium severity issue

Impact:	Medium	Likelihood:	Medium
Target:	`**/*`	Type:	Compute budget

Description

Call `create_program_address` with `BUMP` instead of `find_program_address` on-chain. There's a risk of exceeding a compute budget. We recommend computing the `BUMP` off-chain.

```
#[account(
    constraint = Pubkey::find_program_address(
        &[State::SEED],
        &settings::ID
    ).eq(&(state.key(), state.state_bump)),
    has_one = fee_beneficiar,
)]
pub state: Account<'info, State>,
...
```

Recommendation

```
    #[account(
        constraint = Pubkey::create_program_address(
            &[State::SEED, &[state.state_bump]],
            &settings::ID
        ).eq(&Ok(state.key())),
        has_one = fee_beneficiar,
)]
pub state: Account<'info, State>,
...
```

Go back to Findings Summary

M2: Using API calls instead of SysVar

Medium severity issue

Impact:	Medium	Likelihood:	Medium
Target:	`**/*`	Type:	Sysvar account

Description

As of `solana-program` version `1.6.5`, sysvars can also be accessed without being passed into an entrypoint as an account. It improves security (no need for additional checks) and performance (smaller transactions).

Recommendation

```
self.state.state.check_confirmation_adequacy(
            &Clock::get()?, // instead of `&self.clock`
            self.signature_storage.verify_and_iter(
                signature_storage_key,
                &settings::ID,
                &submission_id.to_bytes(),
            )?,
        )?;
```

Go back to Findings Summary

M3: Extra SEED during checking

Medium severity issue

Impact:	Medium	Likelihood:	Medium
Target:	debridge/src/lib.rs, L#108	Type:	Account constraint

Description

`BridgeFeeInfo::SEED` missing during a creation of `bridge_fee` account at `settings/src/lib.rs [loc: 696]`. Use `BridgeFeeInfo::SEED` in the initialization of the `bridge_fee` account.

Go back to Findings Summary

M4: Comparing bad PubKeys

Medium severity issue

Impact:	Medium	Likelihood:	Medium
Target:	debridge/src/lib.rs, L#113	Type:	Account constraint

Description

Bad PubKeys used for a constraint check. Use `bridge_fee.key()` instead of `chain_support_info.key()`.

Go back to Findings Summary

M5: Badly calculated rent exempt for one day

Medium severity issue

Impact:	Medium	Likelihood:	Medium
Target:	settings/src/lib.rs, L#353-355	Type:	Rent

Description

The comment in the code says that signatures should be stored for one day, but the code calculates an hour_rent. In addition, the calculation uses subtraction instead of division. Use a simple division for calculating `day_rent` instead of `checked_sub()`. Instead of calling two times `checked_sub()`, you should calculate rent for one day as `year_rent / 365`.

```
/// Temporary storage for signatures
/// Stores signatures for one day. Access is obtained by a signed message
bytes
/// [`OraclesKeys`] inside account data
```

Go back to Findings Summary

M6: `BridgeCtx::staking_wallet` bad constraint

Medium severity issue

Impact:	Medium	Likelihood:	Medium
Target:	debridge/src/lib.rs, L#74	Type:	Type

Description

For Mint Bridge it belongs to a `fee_beneficiar`. For Send Bridge it is owned by `Self::mint_authority`. But in `BridgeCtx`, it is checked whether an owner is a `bridge`. In our opinion, the validation of the owner should depend on the type of `bridge`.

Go back to Findings Summary

L1: Typos in several places in code

Low severity issue

Impact:	Low	Likelihood:	Low
Target:	settings/src/state.rs, settings/src/lib.rs	Type:	Code smell

Description

Fix the typos for all occurrences:

- `subsmission` → `submission`
- `transfered` → `transferred`
- `old_reserbed_bps` → `old_min_reserved_bps`

Go back to Findings Summary

L2: Bad naming conventions

Low severity issue

Impact:	Low	Likelihood:	Low
Target:	`**/*`	Type:	Code smell

Description

Use better naming:

- `is_work_now` → `is_working`
- `is_exists` → `exists`

Go back to Findings Summary

L3: NewTypes or type aliases for primitive types

Low severity issue

Impact:	Low	Likelihood:	Low
Target:	`**/*`	Type:	Code smell

Description

Use NewTypes or type aliases for primitive types to increase readability and reliability.

```
pub old_reserbed_bps: u64,
pub bridge_id: [u8; 32],
pub collected_fee: u64,
```

Recommendation

```
pub old_reserbed_bps: BasisPoints,
pub bridge_id: BridgeId,
pub collected_fee: Lamports,
```

Go back to Findings Summary

L4: Missing or Unused code

Low severity issue

Impact:	Low	Likelihood:	Low
Target:	settings/src/event.rs, L#91	Type:	Code smell

Description

Missing Instruction should be added or unused code should be removed:

- Unused event `BridgeFeeInfoUpdated` or there's a missing instruction `update_fee_bridge_info`.

Go back to Findings Summary

L5: Unused accounts

Low severity issue

Impact:	Low	Likelihood:	Low
Target:	debridge/src/lib.rs, settings/src/lib.rs	Type:	Unused account

Description

Unused accounts unnecessarily increase the size of a transaction and should be removed:

`BridgeCtx::spl_token_program`

- Just set it in the `AmountContextBuilder` structure and then never use it.
- It also lacks a check to see if it is a token program.

`BridgeCtx::system_program`

- Also missing a check if it is a system program (account not used so it is not critical).

Go back to Findings Summary

L6: Unconstrained `authority`

Low severity issue

Impact:	Low	Likelihood:	Low
Target:	debridge/src/lib.rs, L#118	Type:	Account constraint

Description

The role of this `authority` is not clear, except that it signs the transaction, it is not used anywhere else. It is not necessary. Better naming it might be appropriate.

Go back to Findings Summary

L7: Using of `ProgramAccount` struct

Low severity issue

Impact:	Low	Likelihood:	Low
Target:	debridge/src/lib.rs, settings/src/lib.rs	Type:	Type

Description

Use `Account` instead of `ProgramAccount` as advised by the Anchor framework author.

Go back to Findings Summary

L8: Add extra optimizations in `Cargo.toml`

Low severity issue

Impact:	Low	Likelihood:	Low
Target:	Cargo.toml	Type:	Compiler settings

Description

Add more optimizations to `Cargo.toml`:

```
[profile.release]
lto = true
opt-level = 's'  // or 'z' or 3
```

Go back to Findings Summary

L9: Use the latest stable Rust version (1.56)

Low severity issue

Impact:	Low	Likelihood:	Low
Target:	`**/*`	Type:	Rust version

Description

Use the latest stable Rust version (1.56).

Go back to Findings Summary

L10: Consider more pedantic `clippy` rules

Low severity issue

Impact:	Low	Likelihood:	Low
Target:		Type:	Lints

Description

Consider using a command line:

```
cargo clippy --workspace -- -W clippy::pedantic -W clippy::nursery -W
clippy::cargo
```

Go back to Findings Summary

Appendix A: Revision 2

Between November 22, 2021 and December 3, 2021, Ackee Blockchain reviewed DeBridge's fixes for the issues identified in this report.

All issues but M1 were correctly fixed. Four new additional issues were found.

The following table summarizes the re-audit.

Fix log

Id	Severity	Impact	Likelihood	Status
H1: `InitSendBridge` computational budget	High	Medium	High	**Fixed**
H2: Custom program errors in `settings` program	High	Medium	High	**Fixed**
H3: The program violates the stack size at runtime	High	Medium	High	**Fixed**
H4: Protocol doesn't collect native fix fee	High	Medium	High	**New**
M1F: Use `create_program_address` instead of `find_program_address`	Medium	Medium	Medium	**Partially Fixed**
M2: Using API calls instead of `SysVar`	Medium	Medium	Medium	**Fixed**
M3: Extra SEED during checking	Medium	Medium	Medium	**Fixed**

Id	Severity	Impact	Likelihood	Status
M4: Comparing bad PubKeys	Medium	Medium	Medium	**Fixed**
M5: Badly calculated rent exempt for one day	Medium	Medium	Medium	**Fixed**
M6: `BridgeCtx::staking_wallet` bad constraint	Medium	Medium	Medium	**Fixed**
L11: Mistakes in documentation	Low	Low	Low	**New**
L12: Ununified approach to system accounts constraint	Low	Low	Low	**New**
L13: Wrongly used range literal	Low	Low	Low	**New**
L1: Typos in several places in code	Low	Low	Low	**Fixed**
L2: Bad naming conventions	Low	Low	Low	**Fixed**
L3: NewTypes or type aliases for primitive types	Low	Low	Low	**Fixed**
L4: Missing or Unused code	Low	Low	Low	**Fixed**
L5: Unused accounts	Low	Low	Low	**Fixed**
L6: Unconstrained `authority`	Low	Low	Low	**Fixed**

Id	Severity	Impact	Likelihood	Status
L7: Using of `ProgramAccount` struct	Low	Low	Low	**Fixed**
L8: Add extra optimizations in `Cargo.toml`	Low	Low	Low	**Fixed**
L9: Use the latest stable Rust version (1.56)	Low	Low	Low	**Fixed**
L10: Consider more pedantic `clippy` rules	Low	Low	Low	**Fixed**

Table 3. Table of fixes

M1F: Use `create_program_address` instead of `find_program_address`

Medium severity issue

Impact:	Medium	Likelihood:	Medium
Target:	M1: Use `create_program_address` instead of `find_program_address`	Type:	Compute budget

Description

There are still several places where `create_program_address` with `BUMP` can be used instead of `find_program_address` on-chain:

- `setting/src/lib.rs [loc: 1039]`
- `setting/src/bridge.rs [loc: 459]`

Go back to Fix log

H4: Protocol doesn't collect native fix fee

High severity issue

Impact:	Medium	Likelihood:	High
Target:	debridge/src/state.rs, L#177	Type:	Application logic

Description

The fix fee is not transferred to the `fee_beneficiar` wallet if fix fees are charged in a native asset. We believe it should be accumulated to `execution_fee` or it should happen in the `take_native_fix_fee` function.

Go back to Fix log

L11: Mistakes in documentation

Low severity issue

Impact:	Low	Likelihood:	Low
Target:	settings/src/lib.rs, debridge/src/lib.rs	Type:	Documentation

Description

Fix mistakes in the documentation:

- `excess` → `minimum`
- `fee_beneficiar` → `stop_tap`
- not used `depth` parameter
- `staking_wallet` doesn't belong to `fee_beneficiar`
- not used `submission_bump`

Go back to Fix log

L12: Ununified approach to system accounts constraint

Low severity issue

Impact:	Low	Likelihood:	Low
Target:		Type:	Account constraint

Description

Unify the system program constraints.

```
#[account(address = system_program::ID)]
system_program: AccountInfo<'info>
```

Recommendation

```
pub system_program: Program<'info, System>
```

Go back to Fix log

L13: Wrongly used range literal

Low severity issue

Impact:	Low	Likelihood:	Low
Target:	signature-verifier/src/lib.rs, L#43	Type:	Code smell

Description

The lower number should be on the left side of the range operator. Alternatively, refactor not to use the range operator with a map if the `depth` param will always be 1.

Go back to Fix log

Appendix B: Complementary audit - Crates

Between July 25, 2022, and August 12, 2022, Ackee Blockchain performed a complementary audit of Crates modules.

The following table summarizes found issues.

Audit crates

Id	Severity	Impact	Likelihood
I1C: Deprecated constant usage	Informational	Low	N/A
W1C: Underflow/overflow	Warning	Low	N/A

Table 4. Table of Findings

I1C: Deprecated constant usage

Low severity issue

Impact:	Low	Likelihood:	N/A
Target:	crates/solana-pubkey/src/lib.rs	Type:	Code smell

Description

In the `try_find_program_address` function on the line 251. There is a construct `let mut bump_seed = [std::u8::MAX]`; the use of `std::u8::MAX` is deprecated and the `u8::MAX` should be used.

Go back to Audit crates

W1C: Underflow/overflow

Impact:	Warning	Likelihood:	N/A
Target:	crates/crypto/src/u256.rs	Type:	Arithmetic

Description

Custom implementation of `add()`, `sub()`, `mul()`, `pow()` for a 256-bit unsigned integer overflow. The program won't panic, but the variable will have a value that probably isn't what you were expecting it to have. The issue cannot be exploited given the current code but could be a security vulnerability if these were to change.

Go back to Audit crates

Appendix C: Revision 2.3

In March, 2023, Ackee Blockchain reviewed DeBridge's fixes for the issues identified in this report.

All issues were correctly fixed.

The following table summarizes the fix-audit.

Fix log

Id	Severity	Impact	Likelihood	Status
H1: `InitSendBridge` computational budget	High	Medium	High	**Fixed**
H2: Custom program errors in `settings` program	High	Medium	High	**Fixed**
H3: The program violates the stack size at runtime	High	Medium	High	**Fixed**
H4: Protocol doesn't collect native fix fee	High	Medium	High	**Fixed**
M1F: Use `create_program_address` instead of `find_program_address`	Medium	Medium	Medium	**Fixed**
M2: Using API calls instead of `SysVar`	Medium	Medium	Medium	**Fixed**
M3: Extra SEED during checking	Medium	Medium	Medium	**Fixed**

Id	Severity	Impact	Likelihood	Status
M4: Comparing bad PubKeys	Medium	Medium	Medium	**Fixed**
M5: Badly calculated rent exempt for one day	Medium	Medium	Medium	**Fixed**
M6: `BridgeCtx::staking_wallet` bad constraint	Medium	Medium	Medium	**Fixed**
L11: Mistakes in documentation	Low	Low	Low	**Fixed**
L12: Ununified approach to system accounts constraint	Low	Low	Low	**Fixed**
L13: Wrongly used range literal	Low	Low	Low	**Fixed**
L1: Typos in several places in code	Low	Low	Low	**Fixed**
L2: Bad naming conventions	Low	Low	Low	**Fixed**
L3: NewTypes or type aliases for primitive types	Low	Low	Low	**Fixed**
L4: Missing or Unused code	Low	Low	Low	**Fixed**
L5: Unused accounts	Low	Low	Low	**Fixed**
L6: Unconstrained `authority`	Low	Low	Low	**Fixed**

Id	Severity	Impact	Likelihood	Status
L7: Using of `ProgramAccount` struct	Low	Low	Low	**Fixed**
L8: Add extra optimizations in `Cargo.toml`	Low	Low	Low	**Fixed**
L9: Use the latest stable Rust version (1.56)	Low	Low	Low	**Fixed**
L10: Consider more pedantic `clippy` rules	Low	Low	Low	**Fixed**
I1C: Deprecated constant usage	Info	N/A	N/A	**Acknowledged**
W1C: Underflow/overflow	Warning	N/A	N/A	**Acknowledged**

Table 5. Table of fixes

Appendix D: InitSentBridge instruction

```
Instruction {
    program_id: BkE5dgkHHhQZZCEjqDsRSLrkfZGYbeSWAjBzRSd2vyzh,
    accounts: [
      { pubkey: DwGXPVMDyqKn8mrdwZmRPExHn7nfMUk1N9vKx99UQBaW, is_signer:
false, is_writable: true },
      { pubkey: 7YxsoxwJN4fWVKLXGCPtKcbZEvYxhnCusV1tMVPnrSsk, is_signer:
false, is_writable: true },
      { pubkey: 3j3LuxfD7Mcfb3EvS3YZxBataL4kXFBxwdKrWtj3T8K8, is_signer:
false, is_writable: true },
      { pubkey: 87so1rooD2gzzyivC824wvJmhDtyk6bkaDeuU4Uq8Qra, is_signer:
false, is_writable: false },
      { pubkey: GRCHaiCmMxS7eYj2E7Qbit5pfUv3rAnZirvYp26QvdbN, is_signer:
false, is_writable: false },
      { pubkey: 11111111111111111111111111111111, is_signer: false,
is_writable: false },
      { pubkey: TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA, is_signer:
false, is_writable: false },
      { pubkey: 3z2acQDyVn5pUSdmVR4VAq9ZTejdxVwXCmX6fKcNX539, is_signer:
true, is_writable: false },
      { pubkey: SysvarRent111111111111111111111111111111111, is_signer:
false, is_writable: false },
      { pubkey: ATokenGPvbdGVxr1b2hvZbsiqW5xWH25efTNsLJA8knL, is_signer:
false, is_writable: false }],
    data: [185, 145, 74, 79, 45, 146, 231, 31, 254, 252]
}
```

Appendix E: Init instructions

E.1. InitFeeBridgeInfo instruction

```
Instruction {
    program_id: BkE5dgkHHhQZZCEjqDsRSLrkfZGYbeSWAjBzRSd2vyzh,
    accounts: [
      { pubkey: AFm6TsQRcDisR5hyFnb6yRZ1B9gzqhahu5HibmG1RAGj, is_signer:
false, is_writable: false },
      { pubkey: GxXs76JYYjdby499AU3Gdy9ftqcyGEPGVJz1tQsppJYJ, is_signer:
false, is_writable: false },
      { pubkey: 7YxsoxwJN4fWVKLXGCPtKcbZEvYxhnCusV1tMVPnrSsk, is_signer:
false, is_writable: true },
      { pubkey: 2Rp8GkdM7sgzMAK47SNpVtTRrzKMEiBrFh8vZEHYK9XA, is_signer:
false, is_writable: true },
      { pubkey: DuFfamGLQEvPeToog1aFUfBohiRVTrGysaPiEkmEVE5s, is_signer:
true, is_writable: false },
      { pubkey: 11111111111111111111111111111111, is_signer: false,
is_writable: false },
      { pubkey: TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA, is_signer:
false, is_writable: false }],
    data: [4, 238, 238, 167, 87, 40, 237, 132, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 101, 255,
10, 0, 0, 0, 0, 0, 0, 0] }
```

E.2. InitChainSupportInfo instruction

```
Instruction {
    program_id: BkE5dgkHHhQZZCEjqDsRSLrkfZGYbeSWAjBzRSd2vyzh,
    accounts: [
      { pubkey: 7YxsoxwJN4fWVKLXGCPtKcbZEvYxhnCusV1tMVPnrSsk, is_signer:
false, is_writable: false },
      { pubkey: DuFfamGLQEvPeToog1aFUfBohiRVTrGysaPiEkmEVE5s, is_signer:
true, is_writable: false },
      { pubkey: DuFfamGLQEvPeToog1aFUfBohiRVTrGysaPiEkmEVE5s, is_signer:
true, is_writable: false },
      { pubkey: J96yUe2LFmw4xYmTBeud8zazxWLYCznCdgayvqzs13Ey, is_signer:
false, is_writable: true },
```

```
      { pubkey: 11111111111111111111111111111111, is_signer: false,
is_writable: false }],
    data: [146, 14, 216, 153, 206, 170, 83, 177, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 101, 255,
1, 100, 0, 0, 0, 0, 0, 0, 0, 232, 3, 0, 0, 0, 0, 0, 0] }
```

E.3. InitDiscountInfo instruction

```
Instruction {
    program_id: BkE5dgkHHhQZZCEjqDsRSLrkfZGYbeSWAjBzRSd2vyzh,
    accounts: [
      { pubkey: 7YxsoxwJN4fWVKLXGCPtKcbZEvYxhnCusV1tMVPnrSsk, is_signer:
false, is_writable: false },
      { pubkey: 5gL3EH72siRwFjpRbwBZbgcAh9mYWajRdnfViTCTgugv, is_signer:
false, is_writable: true },
      { pubkey: 3z2acQDyVn5pUSdmVR4VAq9ZTejdxVwXCmX6fKcNX539, is_signer:
false, is_writable: false },
      { pubkey: 11111111111111111111111111111111, is_signer: false,
is_writable: false },
      { pubkey: DuFfamGLQEvPeToog1aFUfBohiRVTrGysaPiEkmEVE5s, is_signer:
true, is_writable: false }],
    data: [133, 68, 55, 173, 101, 223, 74, 14, 255, 2, 0, 2, 0]
}
```

Appendix F: Send instruction

```
Instruction {
    program_id: 2dhGsWUzy5YKUsjZdLHLmkNpUDAXkNa9MYWsPc4Ziqzy,
    accounts: [
      { pubkey: D3yHckgBcZC65k8sSPc4TnfrwxByn3NizGavFkXaTmwE, is_signer:
false, is_writable: true },
      { pubkey: CUq3ddZMRtzULcBKFELADWDxWBkvM3UfTTE5BaCWvzqE, is_signer:
false, is_writable: false },
      { pubkey: FaKTCzVYj61A37NbX59omCZqFjr3HSUzvJop8TR1bLk9, is_signer:
false, is_writable: false },
      { pubkey: FN87rkMYTnDTxUnAZMu7fQEvePbnL12ieo4FNPStS2kV, is_signer:
false, is_writable: false },
      { pubkey: TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA, is_signer:
false, is_writable: false },
      { pubkey: J96yUe2LFmw4xYmTBeud8zazxWLYCznCdgayvqzs13Ey, is_signer:
false, is_writable: false },
      { pubkey: 2wpXj1q5ELVcF9ufLCAorfvL3jAtk5AkGTWCJFWv1LvZ, is_signer:
false, is_writable: false },
      { pubkey: 5gL3EH72siRwFjpRbwBZbgcAh9mYWajRdnfViTCTgugv, is_signer:
false, is_writable: false },
      { pubkey: 3z2acQDyVn5pUSdmVR4VAq9ZTejdxVwXCmX6fKcNX539, is_signer:
true, is_writable: false },
      { pubkey: TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA, is_signer:
false, is_writable: false },
      { pubkey: SysvarRent111111111111111111111111111111111, is_signer:
false, is_writable: false },
      { pubkey: 11111111111111111111111111111111, is_signer: false,
is_writable: false },
      { pubkey: 7YxsoxwJN4fWVKLXGCPtKcbZEvYxhnCusV1tMVPnrSsk, is_signer:
false, is_writable: false },
      { pubkey: EdKBnC98VgW2EjVqeg9SYKPvJmQn2LRhzLuA4qEXCbSG, is_signer:
false, is_writable: false },
      { pubkey: 4F4sVd5z4cmJqtmaVjumHJ8MtoReCQegtw2xcUZU5nFe, is_signer:
false, is_writable: true },
      { pubkey: 3z2acQDyVn5pUSdmVR4VAq9ZTejdxVwXCmX6fKcNX539, is_signer:
true, is_writable: false },
      { pubkey: 11111111111111111111111111111111, is_signer: false,
is_writable: false }
    ],
    data: [102, 251, 20, 187, 65, 75, 12, 69, 101, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 0,
0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 100, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
}
```

Appendix G: Components diagram - Debridge

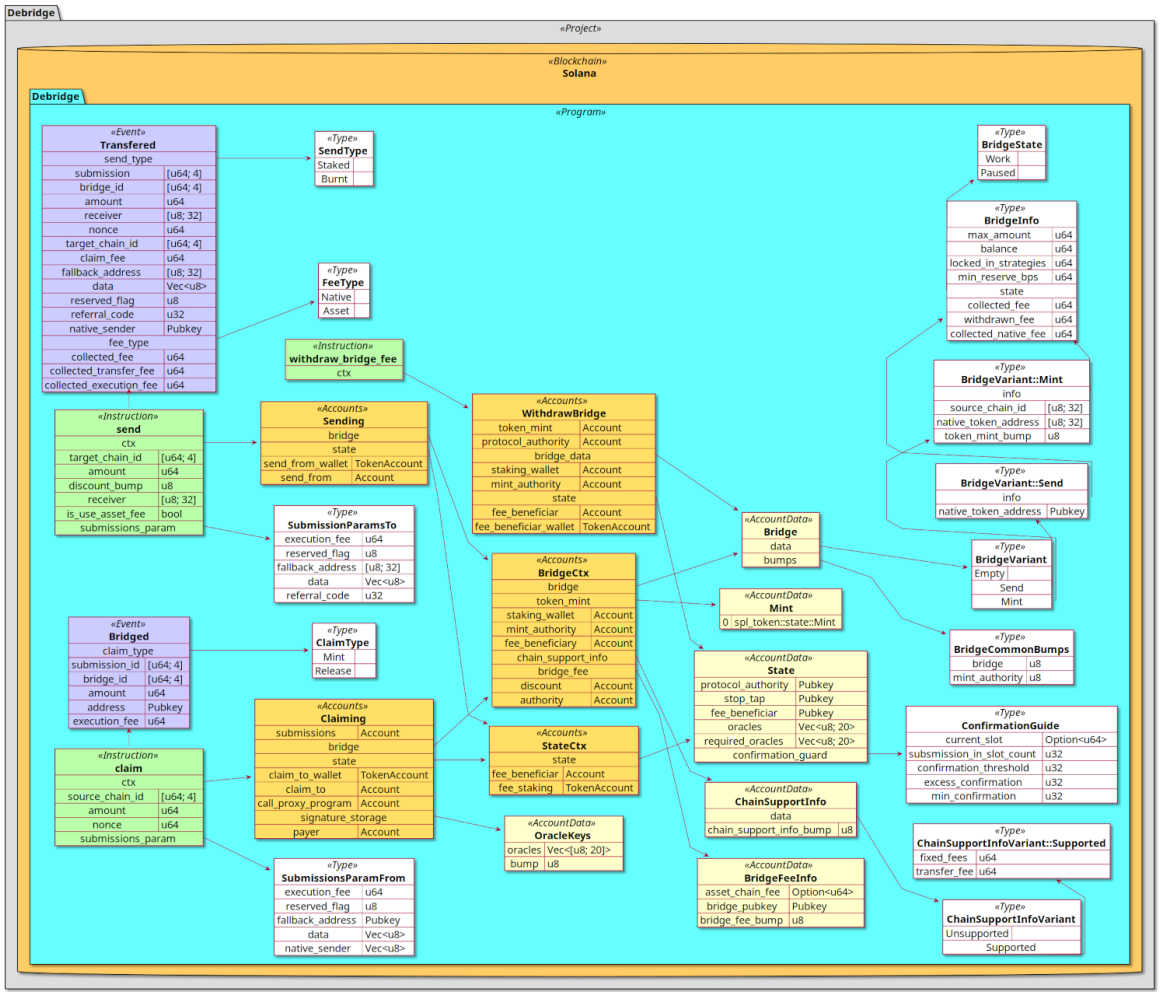


Figure 1. Components diagram - Debridge

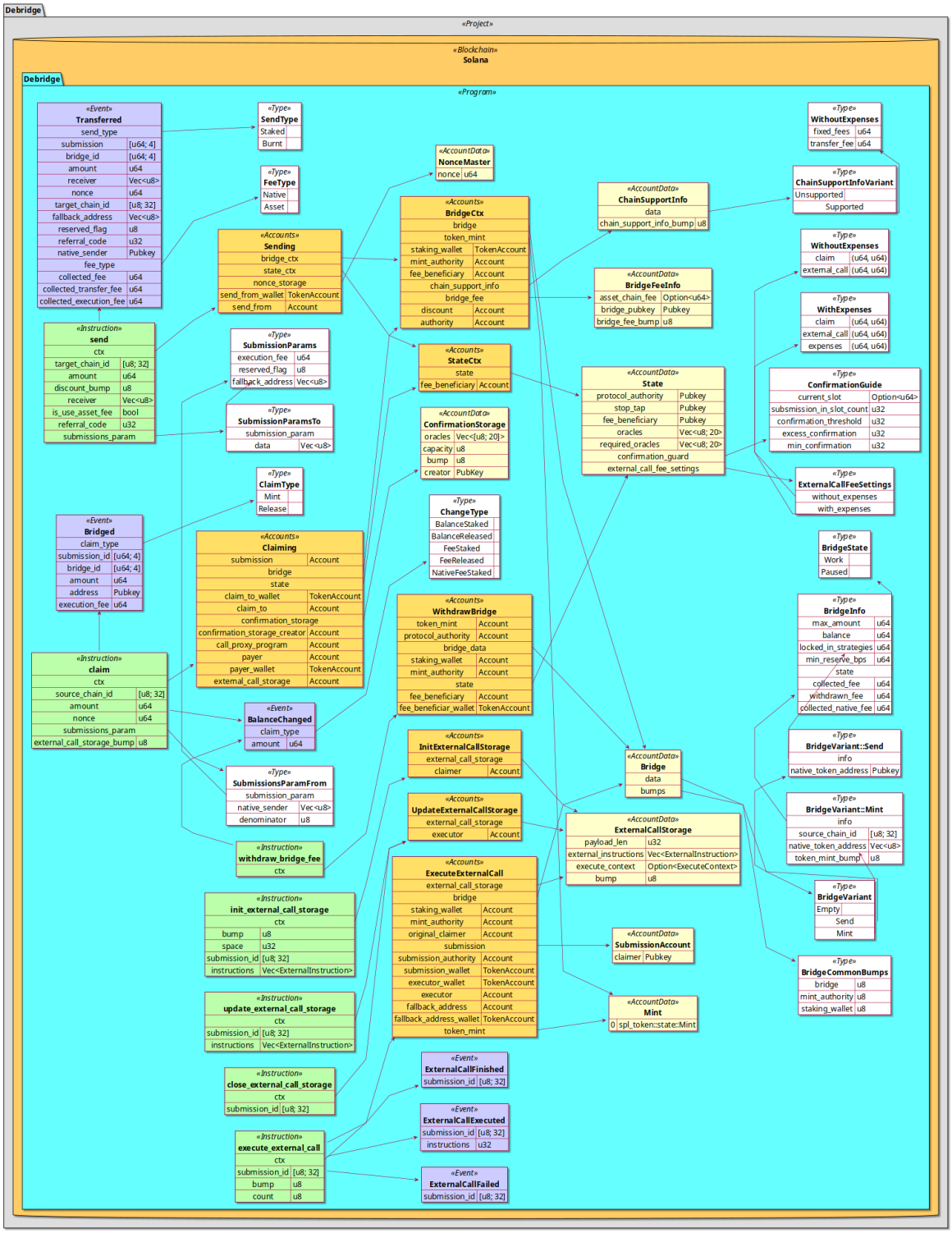


Figure 2. Update rev. 2

Appendix H: Components diagram - Settings

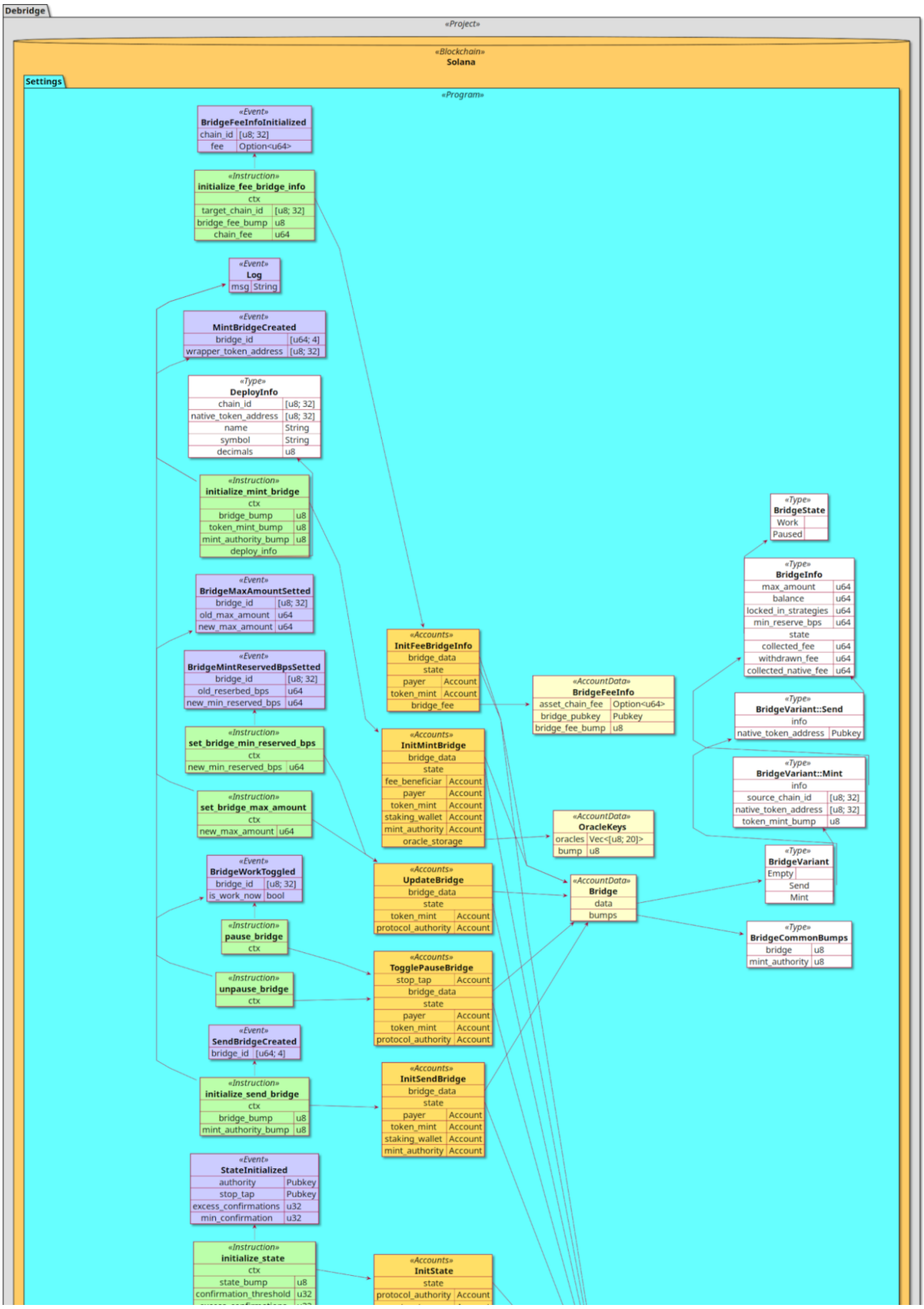


Figure 3. Components diagram - Settings

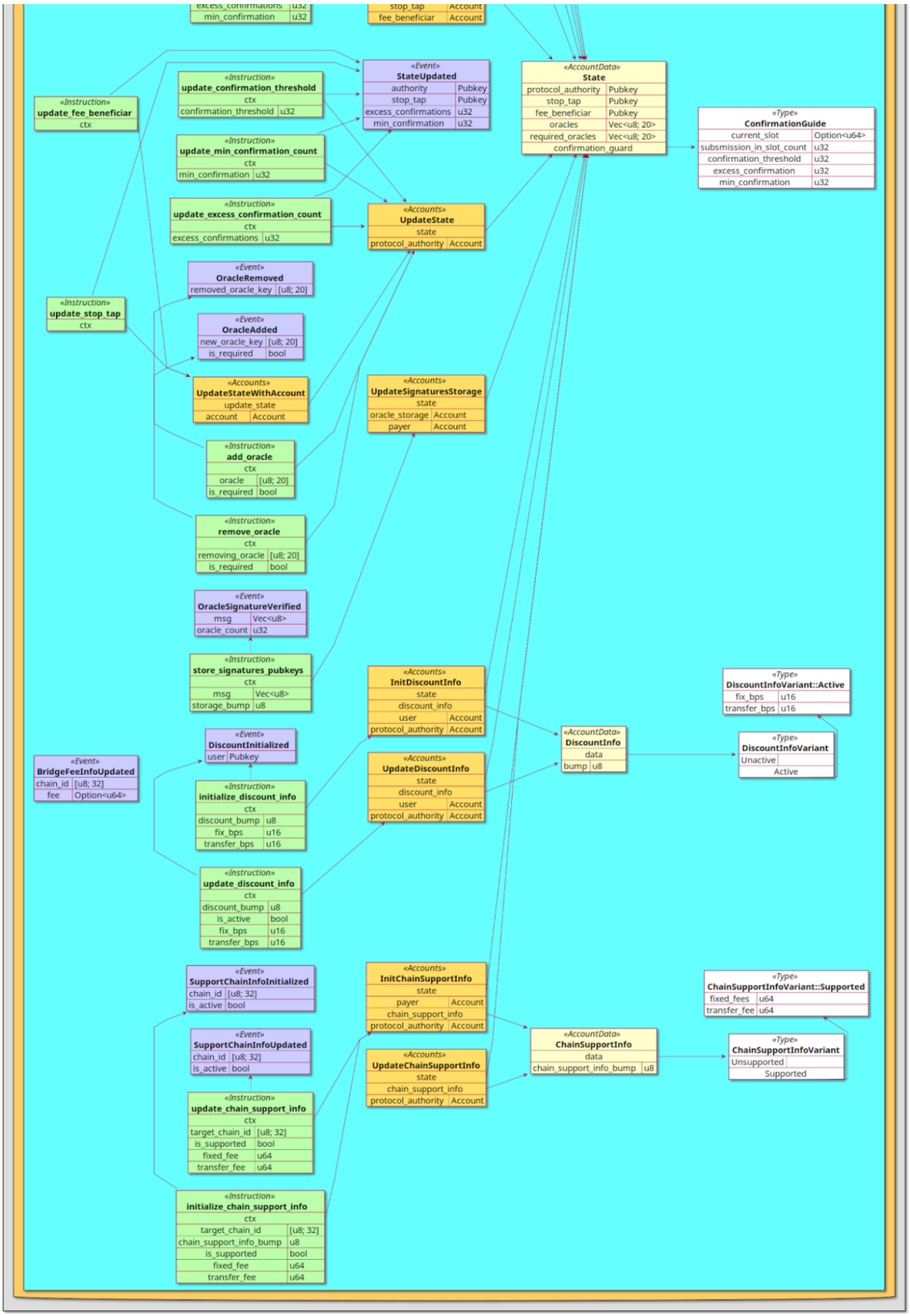


Figure 4. Components diagram - Settings

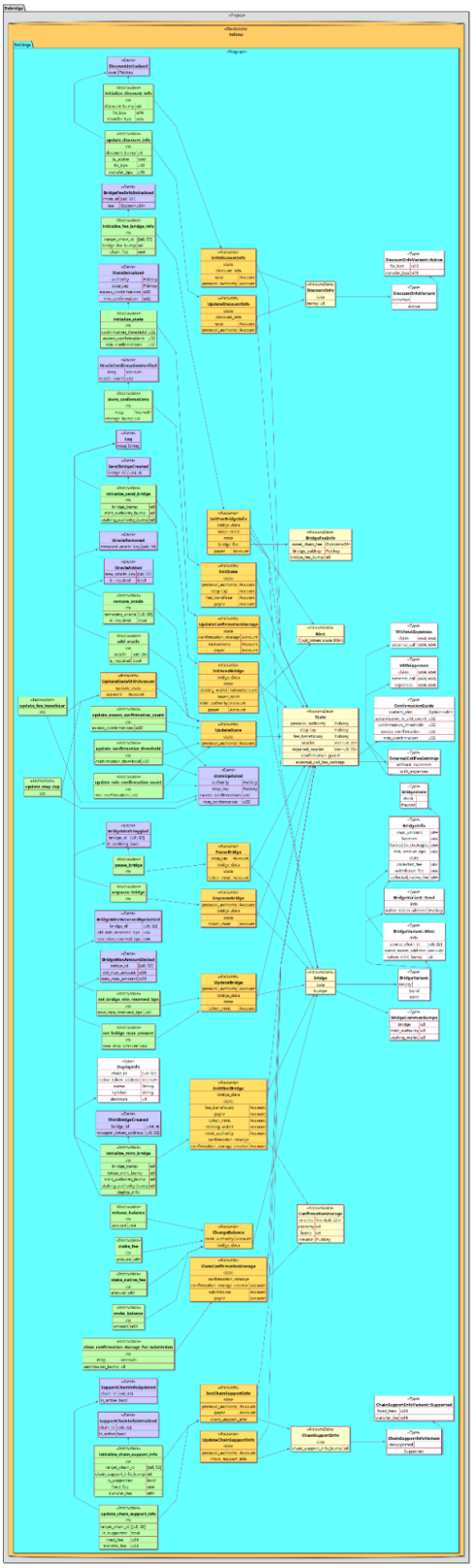


Figure 5. Update rev. 2

Appendix I: How to cite

Please cite this document as:

Ackee Blockchain, DeBridge: Protocol, December 20, 2021.

ackee

Thank You

Ackee Blockchain a.s.


Prague, Czech Republic


hello@ackeeblockchain.com


https://discord.gg/z4KDUbuPxq