certora

Formal Verification Report

SOLANA
FOUNDATION

Token-2022 Extensions

05/24/2024

Prepared for Solana Foundation

Table of content

Project Summary

Protocol Overview

Token Extensions (https://solana.com/solutions/token-extensions) is a new program-level token feature that natively extends token functionality. Token Extensions is at the heart of any Solana application that needs to create and manage tokens. Token Extensions instructions use typically a subset of extensions that can be enabled or disabled based on user-defined options and the account's state.

Project Scope

Project Name	Repository (link)	Latest Commit Hash	Compiler	Platform
Token-2022 Extensions	https://github.com/solana-labs/solana-program-library (public)	260f80928f796fc78c81ef a4dc2a7732665e5a59 (29 Jan 2024)	solana 1.17.2	SBFv1 64-bit

This verification project focuses mainly on formal verification of the following properties:

1. Ownership and well-formedness of accounts.
2. The modifications performed by the extensions on the expected behavior of the accounts are consistent across different SPL instructions.
3. Fee computations satisfy basic correctness properties.

Project Overview

This document describes the specification and verification of the Token Extensions using the Certora Prover. The work was undertaken from 02/01/2024 to 05/15/2024.

The following list of files is included in our scope:

- src/processor.rs
- src/extension/confidential_transfer/processor.rs
- src/extension/transfer_fee/processor.rs
- src/extension/confidential_transfer_fee/processor.rs
- src/extension/cpi_guard/processor.rs
- src/extension/default_account_state/processor.rs
- src/extension/group_member_pointer/processor.rs
- src/extension/group_pointer/processor.rs
- src/extension/interest_bearing_mint/processor.rs
- src/extension/memo_transfer/processor.rs
- src/extension/metadata_pointer/processor.rs
- src/extension/token_group/processor.rs
- src/extension/transfer_hook/processor.rs

We verified all processor functions (with prefix process) in those files except processor functions that support the following instructions:

- InitializeAccount, InitializeAccount2, and InitializeAccount3
- InitializeMultisig and InitializeMultisig2
- InitializeMint and InitializeMint2
- GetAccountDataSize
- Reallocate
- AmountToUiAmount
- UiAmountToAmount

The Certora Prover demonstrated that the implementation of the Solana processor functions above are correct with respect to the formal rules written by the Certora team. During the verification process, the Certora team discovered six low severity issues and two informational issues, as listed below (see Detailed Findings section).

Findings Summary

The table below summarizes the findings of the review, including type and severity details.

Severity	Discovered	Confirmed	Fixed
Critical	0	0	0
High	0	0	0
Medium	0	0	0
Low	6	6	6
Informational	3	3	2
Total			

Severity Matrix

Impact	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Low	Low	Medium
		Low	Medium	High
		Likelihood		

Detailed Findings

ID	Title	Severity	Status
L-01	If CpiGuard is present and enabled (lock_cpi is true), then Transfer, TransferChecked, TransferCheckedWithFee, and Burn are not allowed if signed by the owner.	Low	Fixed https://github.com/solana-labs/solana-program-library/pull/6863/
L-02	Inconsistency in detecting that an account is non-transferable between instructions Transfer, TransferChecked, and TransferCheckedWithFee and ConfidentialTransferInstruction::Transfer	Low	Fixed https://github.com/solana-labs/solana-program-library/pull/6862
L-03	Inconsistent use of MemoTransfer between instructions Transfer, TransferChecked, and TransferCheckedWithFee and ConfidentialTransferInstruction::Transfer	Low	Fixed https://github.com/solana-labs/solana-program-library/pull/6861
L-04	TransferCheckedWithFee succeeds when fee is 0 even if fee extension is not enabled on the mint	Low	Fixed https://github.com/solana-labs/solana-program-library/pull/6860/
L-05	Mixed use of equality operator == and function spl_token_2022::cmp_pubkey to compare public keys	Low	Fixed https://github.com/solana-labs/solana-program-library/pull/6859

L-06	Explicit use of sol_memcmp has a negative effect on performance	Low	Fixed https://github.com/solana-labs/solana-program-library/pull/6859
I-01	The function process_transfer in token-2022/src/processor.rs is used to implement Transfer, TransferChecked, and TransferCheckedWithFee. However, it also succeeds when called with a fee and no mint	Informational	Fixed https://github.com/solana-labs/solana-program-library/pull/6864
I-02	Use of floating points in InterestBearingConfig	Informational	Acknowledged
I-03	calculate_inverse_fee is not an exact inverse of calculate_fee	Informational	Acknowledged https://github.com/solana-labs/solana-program-library/pull/6874

Low Severity Issues

L-01. If CpiGuard is present and enabled (lock_cpi is true), then Transfer, TransferChecked, TransferCheckedWithFee, and Burn are not allowed if signed by the owner.

Description:
Fund decreasing instructions (Transfer, TransferChecked, TransferCheckedWithFee, and Burn) when protected by CpiGuard should not be allowed if signed by the owner. The intention is that they are only allowed if signed by some other account delegate. This property is violated when

1. an account is delegated to itself
2. the account is a permanent delegate of the mint
3. the account is owned by system program or incinerator (in this case, signing authority is not checked and can be arbitrary)

Recommendation:
Revert instruction when it is called in CPI, with enabled CpiGuard, and signed by the source account owner.

Customer’s response:
Acknowledged
https://github.com/solana-labs/solana-program-library/pull/6863/

L-02. Inconsistency in detecting that an account is non-transferable between instructions Transfer, TransferChecked, and TransferCheckedWithFee and ConfidentialTransferInstruction::Transfer

Description:
Regular instructions (Transfer, TransferChecked, and TransferCheckedWithFee) determine that an account is non-transferable by checking for presence of NonTransferableAccount extension in the source account. ConfidentialTransferInstruction::Transfer, instead, checks that the mint has the NonTransferable mint extension.

Recommendation:
This is not an issue at the moment because every account of NonTransferable mint has a NonTransferableAccount extension. We recommend that all instructions use NonTransferableAccount to make the code more uniform and to prevent any deviation in the future should the invariant no longer be maintained.

Customer's response:
Acknowledged
https://github.com/solana-labs/solana-program-library/pull/6862

L-03. Inconsistent use of MemoTransfer between instructions Transfer, TransferChecked, and TransferCheckedWithFee and ConfidentialTransferInstruction::Transfer

Description: In regular instructions (Transfer, TransferChecked, and TransferCheckedWithFee) a self-transfer does not require a memo even if the destination is extended with MemoTransfer while in ConfidentialTransferInstruction::Transfer, self-transfer always does.

Recommendation:
Make regular and confidential instructions behave uniformly with respect to MemoTransfer.

Customer's response:
Acknowledged
https://github.com/solana-labs/solana-program-library/pull/6861

L-04. TransferCheckedWithFee succeeds when fee is 0 even if fee extension is not enabled on the mint

Description: A TransferCheckedWithFee succeeds even if the mint account is not extended with TransferFeeConfig if the instruction argument expected_fees is zero. This makes TransferChecked redundant since it can always be replaced by TransferCheckedWithFee.

Recommendation:

Either not allow TransferCheckedWithFee to succeed when fees are not enabled, or document that this is an expected and allowed behavior.

Customer's response:
Acknowledged
https://github.com/solana-labs/solana-program-library/pull/6860/

L-05. Mixed use of equality operator == and function spl_token_2022::cmp_pubkey() to compare public keys

Description: The struct solana_program::pubkey::Pubkey implements std::cmp::Eq trait via derive macro. Thus, Pubkey should be compared by the builtin equality operator. This results in more efficient code. Hower, spl_token_2022, also defined cmp_pubkey method that implements equality using a less efficient call to sol_memcmp system call.

Recommendation: Change the implementation of cmp_pubkey to use builtin equality operator. Alternatively, deprecate cmp_pubkey and remove all of its uses.

Public keys should be always compared using cmp_pubkey but sometimes they are compared using operator ==.

Affected code:

```
extension/token_group/processor.rs:
  if member_info.key == group_info.key
extension/transfer_fee/processor.rs:
  if account_info.key == destination_account_info.key
extension/confidential_transfer/processor.rs:
   if authority_info.is_signer && *authority_info.key == confidential_transfer_mint_authority
extension/confidential_transfer/processor.rs:
   let is_self_transfer = source_account_info.key == destination_account_info.key;
```

Recommendation: Replace the use of == with cmp_pubkey

Customer’s response:
Acknowledged.
https://github.com/solana-labs/solana-program-library/pull/6859

L-06. Explicit use of sol_memcmp has a negative effect on performance

Description: Solana runtime provides a system call sol_memcmp that implements non-overlapping memory comparison. Rust compiler, and LLVM, provide a builtin intrinsic memcmp for the same operation. Mixing the two, prevents many LLVM optimizations since LLVM treats sol_memcmp as an unknown external function. At the same time, all uses of memcmp are compiled to either multiple word-level compares, or a call to sol_memcmp. The exact choice is controlled by Rust Compilers and LLVM optimization switches.

The lack of these optimizations can have a negative impact on speed and also an increase of the SBF file.
Recommendation: Do not use sol_memcmp, instead use platform independent comparison provided by Rust. This specifically affects comparison of Pubkey, see L-04 for details.

Customer’s response:
Acknowledged.
https://github.com/solana-labs/solana-program-library/pull/6859

Informational Severity Issues

I-01. The function process_transfer in token-2022/src/processor.rs is used to implement Transfer, TransferChecked, and TransferCheckedWithFee. However, it also succeeds when called with a fee and no mint.

Description: The function process_transfer in token-2022/src/processor.rs is used to implement Transfer, TransferChecked, and TransferCheckedWithFee. However, it also succeeds when called with a fee and no mint.

Recommendation: Return an error if the function is called with the last two arguments being None and Some.

Customer’s response:
Acknowledged.
https://github.com/solana-labs/solana-program-library/pull/6864

I-02. Use of floating points in InterestBearingConfig functions

Description: SBF does not support floating point numbers. The type f64 and the corresponding operations in Rust are compiled based on compiler runtime libraries. These libraries may change with each version of the compiler. Thus, the code is not stable and must be re-verified with each change of the compiler

Recommendation: Rework the functionality to not rely on floating point numbers. For example, use a Rust library for fixed point arithmetic instead.

Customer’s response:
Acknowledged without fix.
Customer wrote: we will take that as acceptable -- since the f64 usage is only for UI calculations on mints with interest-bearing config, and not for any internal logic, it's acceptable for the conversion code to be unstable.

I-03. calculate_inverse_fee is not the exact inverse of calculate_fee

Description: The function calculate_inverse_fee is not exactly an inverse operation of calculate_fee. That is, it is not the case that calculate_inverse_fee(x + calculate_fee(x)) == calculate_fee(x).

Recommendation: Document that calculate_inverse_fee is not an exact inverse and instead that only the relationship calculate_fee(x) >= calculate_inverse_fee(x - calculate_fee(x)) holds in order to avoid confusion with the potential users of calculate_inverse_fee.

Customer's response:
Acknowledged.
https://github.com/solana-labs/solana-program-library/pull/6874

Formal Verification

Assumptions and Simplifications

Project General Assumptions

1. We verified all instructions using accounts that contain arbitrary data
2. All accounts are distinct
3. We verified instructions under single owner/delegate
4. We have excluded confidential transfers with split proofs

Code refactoring and explicit summarizations of internal parts of the code

1. Changed the layout of extensions so that each extension starts at a fixed offset. We use the SPL unit tests to validate our changes.
2. Added padding to some data-structures to ensure runtime layout is compatible with the prover.
3. Eliminated pointers to indeterminate stack locations by duplicating relevant code.
4. Stubs for solana system calls (invoke, invoke_signed, sol_get_clock_sysvar, sol_get_stack_depth, sol_get_processed_sibling_instruction, etc.)
5. process_harvest_withheld_tokens_to_mint and process_withdraw_withheld_tokens_from_accounts revert with the first bad-formed account.

General Certora Prover options

1. All loops have been unrolled to three iterations at most (option -bmc 3)

2. The prover assumes (without checking) that within the same program execution, each memory read accesses the same number of bytes than the last memory write (options -optimisticJoins, -optimisticOverlaps)
3. The prover might lift a sequence of pairs of memory loads and stores to memcpy even if it cannot prove statically that the source locations do not overlap with the destination. (option -optimisticMemcpyPromotion)

Furthermore, some rules use the tool option -optimisticMemcmp that uses a world-level model of memcmp. When it is the case, we will include that option in the field “Prover Options".

Verification Notations

Formally Verified	The rule is verified for every state of the contract(s), under the assumptions of the scope/requirements in the rule.
Formally Verified After Fix	The rule was violated due to an issue in the code and was successfully verified after fixing the issue
Violated	A counter-example exists that violates one of the assertions of the rule.

Formal Verification Properties

In this report, we group multiple *assertions* into a single rule. An assertion is a property that the code should satisfy. To ensure that we did not write a *vacuous assertion* (an assertion that is always satisfied regardless of the code), we manually injected a bug in the code for each assertion and proved that each assertion is violated.

src/processor.rs

P-01. Transfer satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_transfer does not revert Accounts = [source, destination, owner] Instruction arguments = [amount]

Assert Name	Status	Description	Prover Options	Link to rule report
src_is_active	*Verified*	*The source account is initialized and not frozen*		link
dst_is_active	*Verified*	*The destination account is initialized and not frozen*		
validate_owner	*Verified*	*The owner account is a signer and either the source delegate or owner*		
non_transferable_account	Verified	*The source cannot have NonTransferableAccount extension*		link

non_transfer_hook	*Verified*	*The source cannot have TransferHookAccount extension*		[*link*]
non_transfer_fee_amount	*Verified*	*The source cannot have TransferFeeAmount extension*		[*link*]
not_cpi	Violated[1]	*If instruction is called via CPI, CpiGuard is present, and lock_cpi is enabled, then the owner is not same as source owner.*		[*link*]

[1] This PR https://github.com/solana-labs/solana-program-library/pull/6724 addressed that problem but missed the case of a permanent delegate. Thus, this assertion still fails after this PR.

P-02. Approve satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_approve does not revert Accounts = [source, delegate, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
src_is_active	Verified	*The source account is initialized and not frozen*		[*link*]
validate_owner	Verified	*The owner account is a signer and the source owner*		

not_cpi	Verified	*If source has CpiGuard and lock_cpi is enabled, then the call was not made in CPI. Approve is prohibited in CPI.*		link

P-03. Revoke satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_revoke does not revert Accounts = [source, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
src_is_active	Verified	*The source account is initialized and not frozen*		link
validate_owner	Verified	*The owner account is a signer and the source delegate or source owner*		

P-04. Revoke satisfies integrity constraints

Status: Verified	Property Assumptions: process_revoke does not revert Accounts = [source, delegate, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
	Verified	*source account does not have anymore a delegate*		[*link*]

P-05. SetAuthority of a token account satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_set_authority does not revert Accounts = [token, owner] Instruction arguments = [authority_type_arg, new_authority_arg]

Assert Name	Status	Description	Prover Options	Link to rule report
account_is_active	Verified	*The token account is initialized and not frozen.*		[*link*]
validate_owner	Verified	*The owner account is a signer. If authority_type_arg is AccountOwner then owner is the token owner.* *If authority_type_arg is CloseAccount then owner is the close authority of the token account*		
not_immutable_owner	Verified	*If authority_type_arg is AccountOwner the token account cannot have extension ImmutableOwner*		[*link*]
disable_locked_cpi	Verified	*if authority_type_arg is AccountOwner and*		

		token has CpiGuard then lock_cpi is always disabled		
not_cpi	Verified	*If authority_type_arg is CloseAccount and token has CpiGuard, and lock_cpi is true then new_authority_arg must be None.*		*link*

P-06. SetAuthority of a mint account satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_set_authority_does not revert Accounts = [mint, owner] Instruction arguments = [authority_type]

Assert Name	Status	Description	Prover Options	Links to rule report
mint_is_active	Verified	*The mint account is initialized*		*link*
validate_owner	Verified	*The owner account is a signer and it matches the proper authority which depends on authority_type*	*-optimisticMemcmp*	*link*

P-07. CloseAccount of a token account satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_close_account does not revert Accounts = [source, destination, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
src_is_initialized	Verified	*The source account is initialized*		[link]
src_neq_dst	Verified	*The source is different from destination*		
validate_owner	Verified	*The owner account is a signer. If the source is not owned by system program or incinerator then owner is the source owner or close authority*		
not_cpi	Verified	*If the source is not owned by system program or incinerator and source owner is not the destination then if source account has CpiGuard then either lock_cpi is false or instruction was not made in CPI.*		[link]

P-08. CloseAccount of a token account satisfies integrity constraints

Status: Verified	Property Assumptions: process_close_account does not revert Accounts = [source, destination, owner]

	Instruction arguments = [expected_decimals]

Assert Name	Status	Description	Prover Options	Link to rule report
src_balance_is_zero	Verified	*If the source is not native then source balance is zero*		[link]()
src_lamports_is_zero	Verified	*source lamports is zero*		
dst_lamports_increase	Verified	*destination lamports is increased by old source lamports*		
src_encrypted_balance_is_zero	Verified	*If source has ConfidentialTransferAccount then encrypted pending and available balances are zero*		[link]()
src_withheld_amount_is_zero	Verified	*If source has TransferFeeAmount then withheld amount is zero*		[link]()
src_encrypted_withheld_amount_is_zero	Verified	*If source has ConfidentialTransferFeeAmount then withheld amount is zero*	*-optimisticMemcmp*	[link]()

P-09. ClosedAccount of a mint account satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_close_account does not revert Accounts = [mint, destination, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint account is initialized*	*-optimisticMemcmp*	*link*
mint_neq_dst	Verified	*The mint is different form destination*		
validate_owner	Verified	*The mint is extended with MintCloseAuthority and owner is a signer and the mint close authority*		

P-10. CloseAccount of a mint account satisfies integrity constraints

Status: Verified	Property Assumptions: process_close_account does not revert Accounts = [mint, destination, owner] Instruction arguments = [expected_decimals]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_expected_decimals	Verified	*The mint decimals field is equal to expected_decimals.*		*link*

mint_supply_is_zero	Verified	*mint supply is zero*		
mint_lamports_is_zero	Verified	*mint lamports is zero*		
dst_lamports_increase	Verified	*destination lamports is increased by mint lamports before the instruction was executed*		

P-11. FreezeAccount satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_toogle_freeze_account does not revert Accounts = [source, mint, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
src_is_active	Verified	*The source account is initialized and not frozen*		link
src_is_not_native	Verified	*The source account is not native*		
mint_is_active	Verified	*The mint account is initialized*		
src_has_mint	Verified	*The mint associated with the source is the mint account*		
validate_owner	Verified	*The owner account is a signer and the mint freeze authority*		

P-12. Instruction FreezeAccount satisfies integrity constraints

Status: Verified	Property Assumptions: process_toogle_freeze_account does not revert Accounts = [source, mint, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
src_is_frozen	Verified	*source is frozen*		link

P-13. ThawAccount satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_toogle_freeze_account does not revert Accounts = [source, mint, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
src_is_initialized	Verified	*The source account is initialized*		link
src_is_frozen	Verified	*The source account is frozen*		
src_is_not_native	Verified	*The source account is not frozen*		

mint_is_active	Verified	*The mint account is initialized*		
src_has_mint	Verified	*The mint associated with the source is the mint account*		
validate_owner	Verified	*The owner account is a signer and the mint freeze authority*		

P-14. Instruction ThawAccount satisfies integrity constraints

Status: Verified	Property Assumptions: process_toogle_freeze_account does not revert Accounts = [source, mint, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
src_is_not_frozen	Verified	*source is not frozen*		[*link*]

P-15. TransferChecked satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_transfer does not revert Accounts = [source, mint, destination, owner]

Assert Name	Status	Description	Prover Options	Link to rule report

src_is_active	Verified	*The source account is initialized and not frozen*	*-optimisticMemcmp*	link
mint_is_active	Verified	*The mint account is initialized*		
src_has_mint	Verified	*The mint associated with the source is the mint account*		
validate_owner	Verified	*The owner account is a signer and the source delegate or owner*		
dst_is_active	Verified	*The destination account is initialized and not frozen*		
dst_has_mint	Verified	*The mint associated with the destination is the mint account*		
not_cpi	Violated[2]	*If instruction is called via CPI, CpiGuard is present, and lock_cpi is true, then the owner is not the same as source owner.*		link
memo_transfer	Verified	*If destination has MemoTransfer extension and requires incoming transfer memos then previous sibling instruction must have the memo*		link
confidential_transfer_allowed	Verified	*If not self transfer and destination has ConfidentialTransferAccount then it must allow confidential transfers*		link

non_transferable_account	Verified	*The source cannot have NonTransferableAccount extension*		link

[2] This PR https://github.com/solana-labs/solana-program-library/pull/6724 addressed that problem but missed the case of a permanent delegate. Thus, this assertion still fails after this PR.

P-16. TransferChecked satisfies integrity constraints

Status: Verified	Property Assumptions: process_transfer does not revert Accounts = [source, mint, destination, owner] Instruction arguments = [expected_decimals, amount]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_expected_decimals	Verified	*The mint decimals is equal to expected_decimals.*		link
self_transfer	Verified	*If self-transfer then both source and destination balances are unmodified*		
self_transfer_native	Verified	*If self-transfer and source is native then both source and destination lamports are unmodified*		
non_self_transfer	Verified	*If not self-transfer then the following holds:* - *source balance is decreased by amount*		

		- *destination balance is increased by amount minus fees*		
non_self_transfer_native	Verified	*If not self-transfer and source is native then the following holds:* - *source lamports is decreased by amount* - *destination lamports is increased by amount*		
delegate_decrease	Verified	*If not self-transfer then source delegate (if any) balance is decreased by amount*	*-optimisticMemcmp*	[link]()
delegate_reset	Verified	*If not self-transfer then if source delegate amount is zero then source has no delegate anymore*		

P-17. ApproveChecked satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_approve does not revert Accounts = [source, mint, delegate, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
src_is_active	Verified	*The source account is initialized and not frozen*		[link]()
mint_is_active	Verified	*The mint account is initialized*		

src_has_mint	Verified	*The mint associated with the source is the mint account*		
validate_owner	Verified	*The owner account is a signer and the source owner*		
not_cpi	Verified	*If source has CpiGuard and lock_cpi is true, then the call was not made in CPI. ApproveChecked is prohibited in CPI.*		*link*

P-18. ApprovedChecked satisfies integrity constraints

Status: Verified	Property Assumptions: process_approve does not revert Accounts = [source, delegate, owner] Instruction arguments = [expected_decimals, amount]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_expected_decimals	Verified	*The mint decimals is equal to expected_decimals.*		*link*
set_delegate	Verified	*delegate is the new source delegate and amount is the new delegate amount*		

P-19. MintTo and MintToChecked satisfy ownership and well-formedness checks

Status: Verified	Property Assumptions: process_mint_to does not revert Accounts = [mint, destination, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint account is initialized*		link
dst_is_active	Verified	*The destination account is initialized and not frozen*		
dst_is_not_native	Verified	*The destination account is not native*		
dst_has_mint	Verified	*The mint associated with the destination is the mint account*		
validate_owner	Verified	*The owner account is a signer and the mint authority*		

P-20. Instruction MintToChecked satisfies integrity constraints

Status: Verified	Property Assumptions: process_mint_to does not revert Accounts = [mint, destination, owner] Instruction arguments = [expected_decimals, amount]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_expected_decimals	Verified	*The mint decimals is equal to expected_decimals.*		[link]()
dst_balance_increase	Verified	*destination balance is increased by amount*		
mint_supply_increase	Verified	*mint supply is increased by amount*		

P-21. Burn and BurnChecked satisfy ownership and well-formedness checks

Status: Verified	Property Assumptions: process_burn does not revert Accounts = [source, mint, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
src_is_active	Verified	*The source account is initialized and not frozen*	*-optimisticMemcmp*	[link]()
src_is_not_native	Verified	*The source account is not native*		
mint_is_active	Verified	*The mint account is initialized*		
src_has_mint	Verified	*The mint associated with the source is the mint account*		

validate_owner	Verified	*If the source is not owned by system program or incinerator then owner is either the permanent delegate, source delegate or the source owner (in this order).*		
not_cpi	Violated	*If source is not owned by system program or incinerator and source has extension CpiGuard with lock_cpi set to true, and called in CPI, then source owner is not the same as owner.*		*link*

P-22. Instruction BurnChecked satisfies integrity constraints

Status: Verified	Property Assumptions: process_burn does not revert Accounts = [source, mint, owner] Instruction arguments = [expected_decimals, amount]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_expected_decimals	Verified	*The mint decimals is equal to expected_decimals.*	*-optimisticMemcmp*	*link*
source_balance_decrease	Verified	*The source balance is decreased by amount*		
mint_supply_decrease	Verified	*The mint supply is decreased by amount*		

delegate_decrease	Verified	*If source is not owned by system program or incinerator then* *source delegate (if any) balance is decreased by amount*		
delegate_reset	Verified	*If source is not owned by system program or incinerator then* *If source delegate amount is zero then source has no delegate anymore*		

P-23. SyncNative satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_sync_native does not revert Accounts = [account]

Assert Name	Status	Description	Prover Options	Link to rule report
account_is_active	Verified	*The account is initialized and not frozen*		[*link*]
account_is_native	Verified	*The account is native*		

P-24. CreateNativeMint satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_create_native_mint does not revert Accounts = [account, mint]

Assert Name	Status	Description	Prover Options	Link to rule report
is_native_mint	*Verified*	*The mint public key is native_mint::id()*		*link*

P-25. WithdrawExcessLamports from a token account satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_withdraw_excess_lamports does not revert Accounts = [source, destination, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
src_is_not_native	Verified	*The source account is not native*		*link*
validate_owner	Verified	*The owner account is a signer and the source owner*		

P-26. WithdrawExcessLamports from a token account satisfies integrity constraints

Status: Verified	Property Assumptions: process_withdraw_excess_lamports does not revert Accounts = [source, destination, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
src_lamports_decrease	Verified	*The source lamports decreases*		[*link*]()
dst_lamports_increase	Verified	*The destination account increases*		

P-27. WithdrawExcessLamports from a mint account satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_withdraw_excess_lamports does not revert Accounts = [mint, destination, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
validate_owner	Verified	*The owner account is a signer and the mint authority*		[*link*]()

P-28. WithdrawExcessLamports from a mint account satisfies integrity constraints

Status: Verified	Property Assumptions: process_withdraw_excess_lamports does not revert Accounts = [mint, destination, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_lamports_decrease	Verified	*The mint lamports decreases*		[link]()
dst_lamports_increase	Verified	*The destination account lamports increases*		

P-29. InitializeCloseAuthority satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: initialize_close_authority does not revert Accounts = [mint]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_not_active	Verified	*The mint base state is not initialized*		[link]()
mint_has_mca	Verified	*The mint account has the MintCloseAuthority*		

P-30. InitializeImmutableOwner satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: initialize_immutable_owner does not revert Accounts = [token]

Assert Name	Status	Description	Prover Options	Link to rule report
token_is_not_initialized	Verified	*The token base state is not initialized*		link
token_has_io	Verified	*The token account has the extension ImmutableOwner*		

P-31. InitializeNonTransferableMint satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: initialize_non_transferable_mint does not revert Accounts = [mint]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_not_active	Verified	*The mint base state is not initialized*		link
mint_has_nt	Verified	*The mint has the NonTransferable extension*		

P-32. InitializePermanentDelegate satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: initialize_permanent_delegate does not revert Accounts = [mint]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_not_active	Verified	*The mint base state is not initialized*		link
mint_has_pd	Verified	*The mint has the PermanentDelegate extension*		

src/extension/confidential_transfer/processor.rs

P-33. Transfer (without split proofs) satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: confidential process_transfer does not revert Accounts = [source, mint, destination, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
src_is_active	Verified	*The source account is initialized and not frozen*		[link]()
src_has_mint	Verified	*The mint associated with the source is the mint account*		
dst_is_active	Verified	*The destination account is initialized and not frozen*		
dst_has_mint	Verified	*The mint associated with the destination is the mint account*		
mint_is_active	Verified	*The mint account is initialized*		
validate_owner	Verified	*The owner account is a signer and the source owner*		
src_has_cta_ext	Verified	*The source account has extension ConfidentialTransferAccount*		[link]()
dst_has_cta_ext	Verified	*The destination account has extension ConfidentialTransferAccount*		

mint_has_ctc_ext	Verified	*The mint account has extension ConfidentialTransferMint*		
memo_transfer	Verified	*If destination has MemoTransfer extension and requires incoming transfer memos then previous sibling instruction must have the memo*		[link]()
non_transferable	Verified	*The mint cannot have NonTransferable extension*		[link]()
has_both_tfc_ext	Verified	*If not self-transfer then if mint has TransferFeeConfig extension then it must have ConfidentialTransferFeeConfig*		[link]()

P-34. Withdraw satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_withdraw does not revert Accounts = [source, mint, _, owner] Instruction arguments = [amount, expected_decimals,...]

Assert Name	Status	Description	Prover Options	Link to rule report
src_is_active	Verified	*The source account is initialized and not frozen*		[link]()
mint_is_active	Verified	*mint is initialized*		
src_has_mint	Verified	*The mint associated with the source is the mint account*		

validate_owner	Verified	*The owner is a signer and the source owner*		
mint_expected_decimals	Verified	*The mint decimals field is equal to expected_decimals.*		
src_has_cta_ext	Verified	*The source account has the extension ConfidentialTransferAccount*		[link]()
non_transferable	Verified	*The mint cannot have NonTransferable extension*		[link]()

P-35. Deposit satisfy ownership and well-formedness checks

Status: Verified	Property Assumptions: process_deposit does not revert Accounts = [token, mint, owner] Instruction arguments = [amount, expected_decimals]

Assert Name	Status	Description	Prover Options	Link to rule report
token_is_active	Verified	*The token account is initialized and not frozen*		[link]()
mint_is_active	Verified	*mint is initialized*		
token_has_mint	Verified	*The mint associated with token account is the mint account*		
validate_owner	Verified	*The owner is a signer and the token account owner*		

token_is_not_native	Verified	*The token account is not a native account*		
mint_expected_decimals	Verified	*The mint decimals field is equal to expected_decimals.*		
token_has_funds	Verified	*The token account amount is greater or equal than argument amount*		
token_has_cta_ext	Verified	*The token account has the extension ConfidentialTransferAccount*		*link*
non_transferable	Verified	*The mint cannot have NonTransferable extension*		*link*

P-36. EmptyAccount satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_empty_account does not revert Accounts = [token, _, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
token_is_initialized	Verified	*The token account is initialized*		*link*
validate_owner	Verified	*The owner account is a signer and the token owner*		
available_balance_is_zero	Verified	*The encrypted available balance is zero*		

pending_balance_is_zero	Verified	*The encrypted pending balance (low and high parts) is zero*		
token_has_cta_ext	Verified	*The token account has the extension ConfidentialTransferAccount*		link

P-37. Approve satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_approve_account does not revert Accounts = [token, mint, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
token_is_initialized	Verified	*The token account is initialized*		link
mint_is_active	Verified	*The mint account (base state) is initialized*		
token_has_mint	Verified	*The mint associated with token account is the mint account*		
validate_owner	Verified	*The owner account is a signer and the confidential transfer mint authority*		
approved_transfer	Verified	*The field approved from ConfidentialTransferAccount is set to true*		
mint_has_ctm_ext	Verified	*The mint account has extension ConfidentialTransferMint*		link

token_has_cta_ext	Verified	*The token account has extension ConfidentialTransferAccount*		

P-38. ConfigureAccount satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_configure_account does not revert Accounts = [token, mint, _, owner, ...]

Assert Name	Status	Description	Prover Options	Link to rule report
token_is_initialized	Verified	*The token account is initialized*		[*link*]
mint_is_active	Verified	*The mint account is initialized*		
token_has_mint	Verified	*The mint associated with token account is the mint account*		
validate_owner	Verified	*The owner account is a signer and the token owner*		
confidential_credits_allowed	Verified	*confidential deposits and transfers are allowed for token*		
non_confidential_credits_allowed	Verified	*non-confidential deposits and transfers are allowed*		
token_has_cta_ext	Verified	*The token account has always ConfidentialTransferAccount extension*		[*link*]

mint_has_ctm_ext	Verified	*The mint account has always ConfidentialTransferMint extension*		
mint_token_have_transfer_fee	Verified	*If mint has TransferFeeConfig then token has always ConfidentialTransferFeeAmount*		

P-39. InitializeMint satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_initialize_mint does not revert Accounts = [mint]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_not_active	Verified	*The mint base state is not initialized*		*link*

P-40. UpdateMint satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_update_mint does not revert Accounts = [mint, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint is initialized*		*link*

validate_owner	Verified	*The owner is a signer and the confidential transfer mint authority*		
mint_has_ctm_ext	Verified	*The mint has the ConfidentialTransferMint extension*		[*link*]()

P-41. ApplyPendingBalance satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_apply_pending_balance does not revert Accounts = [token, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
token_is_initialized	Verified	*The token account is initialized*		[*link*]()
validate_owner	Verified	*The owner account is a signer and the token owner*		
pending_balance_is_zero	Verified	*The encrypted pending balance (low and high parts) is zero*		
pending_balance_credit_counter_is _zero	Verified	*The pending balance credit counter is zero*		
token_has_cta_ext	Verified	*The token account has the ConfidentialTransferAccount extension*		[*link*]()

P-42. EnableConfidentialCredits satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_allow_confidential_credits does not revert Accounts = [token, owner] Instruction argument = [allow_confidential_credits=true]

Assert Name	Status	Description	Prover Options	Link to rule report
token_is_initialized	Verified	*The token account is initialized*		*link*
validate_owner	Verified	*The owner account is a signer and the token owner*		
allow_confidential_credits	Verified	*The flag allow_confidential_credits is set to true*		
token_has_cta_ext	Verified	*The token account has the ConfidentialTransferAmount extension*		

P-43. DisableConfidentialCredits satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_allow_confidential_credits does not revert Accounts = [token, owner] Instruction argument = [allow_confidential_credits=false]

Assert Name	Status	Description	Prover Options	Link to rule report

token_is_initialized	Verified	*The token account is initialized*		link
validate_owner	Verified	*The owner account is a signer and the token owner*		
disallow_confidential_credits	Verified	*The flag allow_confidential_credits is set to false*		
token_has_cta_ext	Verified	*The token account has the ConfidentialTransferAmount extension*		

P-44. EnableNonConfidentialCredits satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_allow_non_confidential_credits does not revert Accounts = [token, owner] Instruction argument = [allow_non_confidential_credits=true]

Assert Name	Status	Description	Prover Options	Link to rule report
token_is_initialized	Verified	*The token account is initialized*		link
validate_owner	Verified	*The owner account is a signer and the token owner*		
allow_non_confidential_credits	Verified	*The flag allow_non_confidential_credits is set to true*		

token_has_cta_ext	Verified	*The token account has the ConfidentialTransferAmount extension*		

P-45. DisableNonConfidentialCredits satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_allow_non_confidential_credits does not revert Accounts = [token, owner] Instruction argument = [allow_non_confidential_credits=false]

Assert Name	Status	Description	Prover Options	Link to rule report
token_is_initialized	Verified	*The token account is initialized*		link
validate_owner	Verified	*The owner account is a signer and the token owner*		
disallow_non_confidential_credits	Verified	*The flag allow_non_confidential_credits is set to false*		
token_has_cta_ext	Verified	*The token account has the ConfidentialTransferAmount extension*		

src/extension/transfer_fee/processor.rs

P-46. InitializeTransferFeeConfig satisfies ownership and well-formedness checks	
Status: Verified	Property Assumptions: process_initialize_transfer_fee_config does not revert Accounts = [mint] Instructions arguments = [transfer_fee_config_authority_arg, withdraw_withheld_authority_arg]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_not_initialized	Verified	*The mint base state is not initialized*		link
set_transfer_fee_config_authority	Verified	*The TransferFeeConfig field transfer_fee_config_authority is set transfer_fee_config_authority_arg*		
set_withdraw_withheld_authority	Verified	*The TransferFeeConfig field withdraw_withheld_authority is set to withdraw_withheld_authority_arg*		

P-47. TransferCheckedWithFee satisfies ownership and well-formedness checks
Status: Verified Note: all assertions related to ownership and well-formedness have been already proven in TransferChecked

P-48. WithdrawWithheldTokensFromMint satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: does not revert Accounts = [mint, destination, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint is initialized*		[link]()
dst_is_active	Verified	*The destination account is initialized and not frozen*		
dst_has_mint	Verified	*The mint associated with destination account is the mint account*		
validate_owner	Verified	*The owner account is a signer and the withdraw_witheld_authority is from mint TransferFeeConfig extension*		
withheld_amount_is_zero	Verified	*withheld_amount from mint TransferFeeConfig extension is zero*		
dst_balance	Verified	*The destination amount is increased by old withheld_amount from mint TransferFeeConfig*		

mint_has_tfc_ext	Verified	*The mint account has the TransferFeeConfig extension*		link
dst_has_tfa_ext	Verified	*The destination account has the TransferFeeAmount extension*		

P-49. WithdrawWithheldTokensFromAccounts satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_withdraw_withheld_tokens_from_accounts does not revert **The number of source accounts to withdraw from is fixed to 3** Accounts = [mint, destination, owner, source1, source2, source3]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint account is initialized*		link
dst_is_active	Verified	*The destination is initialized and not frozen*		
dst_has_mint	Verified	*The mint associated with destination account is the mint account*		
sources_are_initialized	Verified	*Each source account is initialized*		
sources_have_mint	Verified	*The mint associated with source accounts is the mint account*		

validate_owner	Verified	*The owner account is a signer and the withdraw withheld authority from mint TransferFeeConfig*		
source_withheld_amount_is_zero	Verified	*withheld_amount from each source TransferFeeAmount is zero*		[link]()
dst_withheld_amount_increases	Verified	*destination amount cannot decrease*		
mint_has_tfc_ext	Verified	*The mint account has the TransferFeeConfig extension*		[link]()
sources_have_tfa_ext	Verified	*The source accounts have the TransferFeeAmount extension*		

P-50. HarvestWithheldTokensToMint satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_harvest_withheld_tokens_to_mint does not revert **The number of source accounts to harvest from is fixed to 3** Accounts = [mint, source1, source2, source3]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint account is initialized*		[link]()

sources_are_initialized	Verified	*The source accounts are initialized*		
sources_have_mint	Verified	*The mint associated with source accounts is the mint account*		
source_withheld_amount_is_zero	Verified	*withheld_amount from each source TransferFeeAmount is zero*		
mint_withheld_amount_increases	Verified	*Withheld_amount from the mint TransferFeeConfig extension cannot decrease*		
mint_has_tfc_ext	Verified	*The mint account has the TransferFeeConfig extension*		*link*
sources_have_tfa_ext	Verified	*The source accounts have the TransferFeeAmount extension*		

.

P-51. SetTransferFee satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_set_transfer_fee does not revert Accounts = [mint, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint account is initialized*		*link*

validate_owner	Verified	*The owner account is a signer and the transfer fee config authority*		
mint_has_tfc_ext	Verified	*The mint account has the TransferFeeConfig extension*		*link*

src/extension/confidential_transfer_fee/processor.rs

P-52. InitializeConfidentialTransferFeeConfig satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_initialize_confidential_transfer_fee_config does not revert Accounts = [mint] Instruction arguments = [authority_arg, withdraw_withheld_authority_elgamal_pubkey_arg]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_not_initialized	Verified	*The mint base state is not initialized*		link
withheld_amount_is_zero	Verified	*ConfidentialTransferFeeConfig withheld_zero is initialized to zero*		
set_authority	Verified	*ConfidentialTransferFeeConfig field authority is set to authority_arg*		
set_withdraw_withheld_authority	Verified	*ConfidentialTransferFeeConfig field withdraw_withheld_authority_elgamal_pubkey is set to withdraw_withheld_authority_elgamal_pubkey_arg*		

P-53. WithdrawWithheldTokensFromMint satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_withdraw_withheld_tokens_from_min does not revert Accounts = [mint, destination, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint account is initialized*		[link](#)
dst_is_active	Verified	*The destination is initialized and not frozen*		
dst_has_mint	Verified	*The mint associated with destination account is the mint account*		
validate_owner	Verified	*The owner is a signer and the withdraw_withheld_authority (from mint TransferFeeConfig extension)*		
withheld_amount_is_zero	Verified	*withheld_amount from mint ConfidentialTransferFeeConfig extension is zero*		
mint_has_tfc_ext	Verified	*The mint has TransferFeeConfig extension*		[link](#)
mint_has_ctfc_ext	Verified	*The mint has ConfidentialTransferFeeConfig*		
dst_has_cta_ext	Verified	*The destination has the ConfidentialTransferAmount*		

P-54. WithdrawWithheldTokensFromAccounts satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_withdraw_withheld_tokens_from_accounts does not revert **The number of source accounts to withdraw from is fixed to 3** Accounts = [mint, destination, _, owner, source1, source2, source3]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint is initialized*		link
dst_is_active	Verified	*The destination is initialized and not frozen*		
dst_has_mint	Verified	*The mint associated with destination account is the mint account*		
validate_owner	Verified	*The owner account is a signer and the withdraw_withheld_authority (mint TransferFeeConfig)*		
sources_are_initialized	Verified	*Each source account is initialized*		
sources_have_mint	Verified	*The mint associated with source accounts is the mint account*		
source_withheld_amount_is_zero	Verified	*withheld_amount from each source ConfidentialTransferFeeAmount is zero*		
mint_has_tfc_ext	Verified	*The mint has TransferFeeConfig extension*		link

mint_has_ctfc_ext	Verified	*The mint has ConfidentialTransferFeeConfig*		
sources_have_ctfa_ext	Verified	*Each source account has ConfidentialTransferFeeAmount extension*		

P-55. HarvestWithheldTokensToMint satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_harvest_withheld_tokens_to_mint does not revert **The number of source accounts to harvest from is fixed to 3** Accounts = [mint, source1, source2, source3]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint account is initialized*		link
sources_are_initialized	Verified	*The source accounts are initialized*		
sources_have_mint	Verified	*The mint associated with source accounts is the mint account*		
source_withheld_amount_is_zero	Verified	*withheld_amount from each source ConfidentialTransferFeeAmount is zero*		
mint_has_ctfc_ext	Verified	*The mint account has the ConfidentialTransferFeeConfig extension*		link

sources_have_ctfa_ext	Verified	*The source accounts have the ConfidentialTransferFeeAmount extension*		

P-56. EnableHarvestToMint satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_enable_harvest_to_mint does not revert Accounts = [mint, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint is initialized*		[link]()
validate_owner	Verified	*The owner is a signer and the authority from mint ConfidentialTransferFeeConfig extension*		
enable_harvest_to_mint	Verified	*The field harvest_to_mint_enabled from mint ConfidentialTransferFeeConfig extension is true*		
mint_has_ctfc_ext	Verified	*The mint has the ConfidentialTransferFeeConfig extension*		

P-57. DisableHarvestToMint satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_disable_harvest_to_mint does not revert Accounts = [mint, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint is initialized*		[link]()
validate_owner	Verified	*The owner is a signer and the authority from mint ConfidentialTransferFeeConfig extension*		
disable_harvest_to_mint	Verified	*The field harvest_to_mint_enabled from mint ConfidentialTransferFeeConfig extension is false*		
mint_has_ctfc_ext	Verified	*The mint has the ConfidentialTransferFeeConfig extension*		

src/extension/cpi_guard/processor.rs

P-58. Enable satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_toggle_cpi_guard does not revert Accounts = [token, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
token_is_initialized	Verified	*The token account is initialized*		[*link*]
validate_owner	Verified	*The owner account is a signer and the token owner*		
not_cpi	Verified	*The instruction was not made in CPI.*		
enable_lock_cpi	Verified	*The CpiGuard field lock_cpi is enabled*		
token_has_cpi_ext	Verified	*The token account is extended with CpiGuard*		

P-59. Disable satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_toggle_cpi_guard does not revert

	Accounts = [token, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
token_is_initialized	Verified	*The token account is initialized*		*link*
validate_owner	Verified	*The owner account is a signer and the token owner*		
not_cpi	Verified	*The instruction was not made in CPI*		
disable_lock_cpi	Verified	*The CpiGuard field lock_cpi is disabled*		
token_has_cpi_ext	Verified	*The token account is extended with CpiGuard*		

src/extension/default_account_state/processor.rs

P-60. Initialize satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_initialize_default_account_state does not revert Accounts = [mint] Instruction arguments = [state_arg]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_not_initialized	Verified	*The mint base state is not initialized*		[link]()
set_state	Verified	*The field state from DefaultAccountState extension is set to state_arg*		

P-61. Update satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_update_default_account_state does not revert Accounts = [mint, owner] Instruction arguments = [state_arg]

Assert Name	Status	Description	Prover Options	Link to rule report

mint_is_active	Verified	*The mint is initialized*		[link]()
validate_owner	Verified	*The owner account is a signer and the mint freeze authority*		
state_is_not_uninitialized	Verified	*The field state from DefaultAccountState cannot be Uninitialized*		
set_state	Verified	*The field state from DefaultAccountState extension is set to state_arg*		
mint_has_das_ext	Verified	*The mint has the extension DefaultAccountState*		

src/extension/group_member_pointer/processor.rs

P-62. Initialize satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_initialize_group_member_pointer does not revert Accounts = [mint] Instruction arguments = [authority_arg, member_address_arg]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_not_initialized	Verified	*The mint base state is not initialized*	*-optimisticMemcmp*	*link*
default_key	Verified	*Either authority_arg or member_address_arg is not a default public key*		

P-63. Update satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_update_group_member_pointer does not revert Accounts = [mint, owner] Instruction arguments = [member_address_arg]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint is initialized*	*-optimisticMemcmp*	*link*

validate_owner	Verified	*The owner account is a signer and the group member pointer authority*		
set_member_address	Verified	*The field member_address from GroupMemberPointer extension is set to member_address_arg*		
mint_has_gmp_ext	Verified	*The mint has the extension GroupMemberPointer*		

src/extension/group_pointer/processor.rs

P-64. Initialize satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_initialize_group_pointer does not revert Accounts = [mint] Instruction arguments = [authority_arg, group_address_arg]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_not_initialized	Verified	*The mint base state is not initialized*	*-optimisticMemcmp*	*link*
default_key	Verified	*Either authority_arg or group_address_arg is not a default public key*		

P-65. Update satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_update_group_pointer does not revert Accounts = [mint, owner] Instruction arguments = [group_address_arg]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint is initialized*	*-optimisticMemcmp*	*link*

validate_owner	Verified	*The owner account is a signer and the group pointer authority*		
set_group_address	Verified	*The field group_address from GroupPointer extension is set to group_address_arg*		
mint_has_gp_ext	Verified	*The mint has the extension GroupPointer*		

src/extension/interest_bearing_mint/processor.rs

P-66. Initialize satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_initialize_interest_bearing_mint not revert Accounts = [mint] Instruction arguments = [state_arg]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_not_initialized	Verified	*The mint base state is not initialized*		[link]()

P-67. Update satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_update_rate does not revert Accounts = [mint, owner] Instruction arguments = [state_arg]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint is initialized*		[link]()
validate_owner	Verified	*The owner account is a signer and the*		

		InterestBearingConfig authority		
mint_has_ibc_ext	Verified	*The mint has the extension InterestBearingConfig*		

src/extension/memo_transfer/processor.rs

P-68. Enable satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_toggle_required_memo_transfers does not revert Accounts = [token, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
token_is_initialized	Verified	*The token account is initialized*		[link]()
validate_owner	Verified	*The owner account is a signer and the token owner*		
enable_memo_transfer	Verified	*The field require_incoming_transfer_memos from MemoTransfer extension is set to true*		
token_has_mt_ext	Verified	*The token account has the extension MemoTransfer*		

P-69. Disable satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_toggle_required_memo_transfers does not revert Accounts = [token, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
token_is_initialized	Verified	*The token account is initialized*		[link]()
validate_owner	Verified	*The owner account is a signer and the token owner*		
disable_memo_transfer	Verified	*The field require_incoming_transfer_memos from MemoTransfer extension is set to false*		
token_has_mt_ext	Verified	*The token account has the extension MemoTransfer*		

src/extension/metadata_pointer/processor.rs

P-70. Initialize satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_initialize_metadata_pointer does not revert Accounts = [mint] Instruction arguments = [authority_arg, metadata_address_arg]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_not_initialized	Verified	*The mint base state is not initialized*	*-optimisticMemcmp*	*link*
default_key	Verified	*Either authority_arg or metadata_address_arg is not a default public key*		

P-71. Update satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_update_metadata_pointer does not revert Accounts = [mint, owner] Instruction arguments = [metadata_address_arg]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint is initialized*	*-optimisticMemcmp*	*[link]*
validate_owner	Verified	*The owner account is a signer and the metadata pointer authority*		
set_metadata_address	Verified	*The field metadata_address from MetadataPointer extension is set to metadata_address_arg*		
mint_has_mp_ext	Verified	*The mint has the extension MetadataPointer*		

src/extension/token_group/processor.rs

P-72. InitializeGroup satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_initialize_group does not revert Accounts = [group, mint, owner]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint is initialized*		[link]()
group_eq_mint	Verified	*The group account is the mint*		
validate_owner	*Verified*	*The owner account is a signer and the mint authority*		

P-73. UpdateGroupAuthority satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_update_group_authority does not revert Accounts = [group, authority] Instruction argument = [new_authority_arg]

Assert Name	Status	Description	Prover Options	Link to rule

				report
mint_is_active	Verified	*The group account is initialized*	*-optimisticMemcmp*	[*link*]
validate_owner	Verified	*The authority account is a signer and the update_authority from group TokenGroup extension*		
set_authority	Verified	*The update_authority field from group TokenGroup extension is set to new_authority_arg*		
mint_has_tg_ext	Verified	*The mint account has the TokenGroup extension*		

P-74. InitializeMember satisfies ownership and well-formedness checks	
Status: Verified	Property Assumptions: process_initialize_member does not revert Accounts = [member, member_mint, member_mint_authority, group, group_update_authority]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The member_mint account is initialized*	*-optimisticMemcmp*	[*link*]
mint_is_active	Verified	*The group account is initialized*		
member_eq_member_mint	*Verified*	*The member account is equal to the member_mint account*		

member_neq_group	*Verified*	*The member account cannot be equal to the group account*		
validate_owner	*Verified*	*The member_mint_authority account is the member_mint authority*		
group_authority	*Verified*	*The group_update_authority is the group update authority*		
added_member	Verified	*The member has been added to the group:* *1) the field group (resp. mint) from member TokenGroupMember is the public key of the group (resp. mint) account.* *2) the size of the group is increased by one (in TokenGroup extension from group)*		
member_has_tgm_ext	*Verified*	*The member account has the TokenGroupMember extension*		
member_mint_has_gmp_ext	Verified	*The member_mint account has the GroupMemberPointer extension*		[*link*]
group_has_tg_ext	Verified	*The group account has the TokenGroup extension*		

P-75. UpdateGroupMaxSize satisfies ownership and well-formedness checks	
Status: Verified	Property Assumptions: process_update_group_authority does not revert Accounts = [group, owner] Instruction argument = [max_size_arg]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The group account is initialized*		[*link*]()
validate_owner	Verified	*The owner account is a signer and the update_authority of the TokenGroup extension from group*		
set_max_size	*Verified*	*The field max_size from TokenGroup extension in group is updated to max_size_arg*		
group_has_tg_ext	Verified	*The group account has the TokenGroup extension*		

src/extension/transfer_hook/processor.rs

P-76. Initialize satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_initialize_transfer_hook does not revert Accounts = [mint] Instruction arguments = [authority_arg, hook_address_arg]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_not_initialized	Verified	*The mint base state is not initialized*	*-optimisticMemcmp*	*link*
default_key	Verified	*Either authority_arg or hook_address_arg is not a default public key*		

P-77. Update satisfies ownership and well-formedness checks

Status: Verified	Property Assumptions: process_update_rate does not revert Accounts = [mint, owner] Instruction arguments = [program_id_arg]

Assert Name	Status	Description	Prover Options	Link to rule report
mint_is_active	Verified	*The mint is initialized*	*-optimisticMemcmp*	*link*

validate_owner	Verified	*The owner account is a signer and the TransferHook authority*		
set_program	Verified	*The field program_id from TransferHook is set to program_id_arg*		
mint_has_th_ext	Verified	*The mint has the extension TransferHook*		

Calculation of fees

We use tfbps as an abbreviation of transfer_fee_basis_points.

P-78. Function calculate_inverse_fee is the inverse of calculate_fee

Status: Verified	Property Assumptions:

Assert Name	Status	Description	Assumptions	Prover Options	Link to rule report
inverse	Verified	*calculate_inverse_fee(x- calculate_fee(x)) <= calculate_fee(x)*	*maximum_fee <= 9_000*		link

The assumption maximum_fee <= 9_000 is needed to avoid the solver timeout.

P-79. Function calculate_fee satisfies integrity constraints

Status: Verified	Property Assumptions:

Assert Name	Status	Description	Prover Options	Link to rule report
additivity	Verified	*If tfbps <= MAX_FEE_BASIS_POINTS then calculate_fee(x) + calculate_fee(y) >= calculate_fee(x+y)*		link
monotonicity	Verified	*If tfbps <= MAX_FEE_BASIS_POINTS and x > y then calculate_fee(x) > calculate_fee(y)*		link

non_zero	Verified	*If 0 < tfbps <= MAX_FEE_BASIS_POINTS and* *maximum_fee > 0 and* *x > 0 then* *calculate_fee(x) > 0*		link

P-80. Function calculate_pre_fee satisfies integrity constraints	
Status: Verified	Property Assumptions:

Assert Name	Status	Description	Prover Options	Link to rule report
maximum	Verified	*If tfbps <= MAX_FEE_BASIS_POINTS then* *calculate_pre_fee_amount(x) <= x +* *maximum_fee*		link
zero	Verified	*If tfbps <= MAX_FEE_BASIS_POINTS and* *x==0 then* *calculate_pre_fee_amount(x) ==0*		
increasing	Violated[3]	*If 0 < **tfbps < MAX_FEE_BASIS_POINTS** and* *x>0 then* *calculate_pre_fee_amount(x) > 0*		link

[3] The assertion is **not** satisfied if tfbps == MAX_FEE_BASIS_POINTS. This has been fixed on May 8, 2024 by this pull request https://github.com/solana-labs/solana-program-library/pull/6704

Other properties

P-81. Data representation for Account and Mint are disjoint

Status: Verified	Property Assumptions:

Assert Name	Status	Description	Prover Options	Link to rule report
account_not_mint	Verified	*There is no data that can be interpreted as both an Account and Mint*		link

P-82. Consistency between checked and unchecked transfer without fees

Status: Verified	Property Assumptions:

Assert Name	Status	Description	Prover Options	Link to rule report
consistency_checked_unchecked	Verified	*If checked transfer reverts then unchecked transfer always revert*		link

P-83. Consistency of checked transfer with fees

Status: Verified	Property Assumptions:

Assert Name	Status	Description	Prover Options	Link to rule report
consistency_transfer_fees	Verified[6]	*If mint does not have TransferFeeConfig extension then checked transfer with* ***expected_fees > 0*** *always reverts*		[link]()

[6] This assertion is not satisfied if expected_fees == 0

Disclaimer

The Certora Prover takes a contract and a specification as input and formally proves that the contract satisfies the specification in all scenarios. Notably, the guarantees of the Certora Prover are scoped to the provided specification and the Certora Prover does not check any cases not covered by the specification.

Even though we hope this information is helpful, we provide no warranty of any kind, explicit or implied. The contents of this report should not be construed as a complete guarantee that the contract is secure in all dimensions. In no event shall Certora or any of its employees be liable for any claim, damages, or other liability, whether in an action of contract, tort, or otherwise, arising from, out of, or in connection with the results reported here.

About Certora

Certora is a Web3 security company that provides industry-leading formal verification tools and smart contract audits. Certora's flagship security product, Certora Prover, is a unique SaaS product that automatically locates even the most rare & hard-to-find bugs on your smart contracts or mathematically proves their absence. The Certora Prover plugs into your standard deployment pipeline. It is helpful for smart contract developers and security researchers during auditing and bug bounties.

Certora also provides services such as auditing, formal verification projects, and incident response.