Zellic

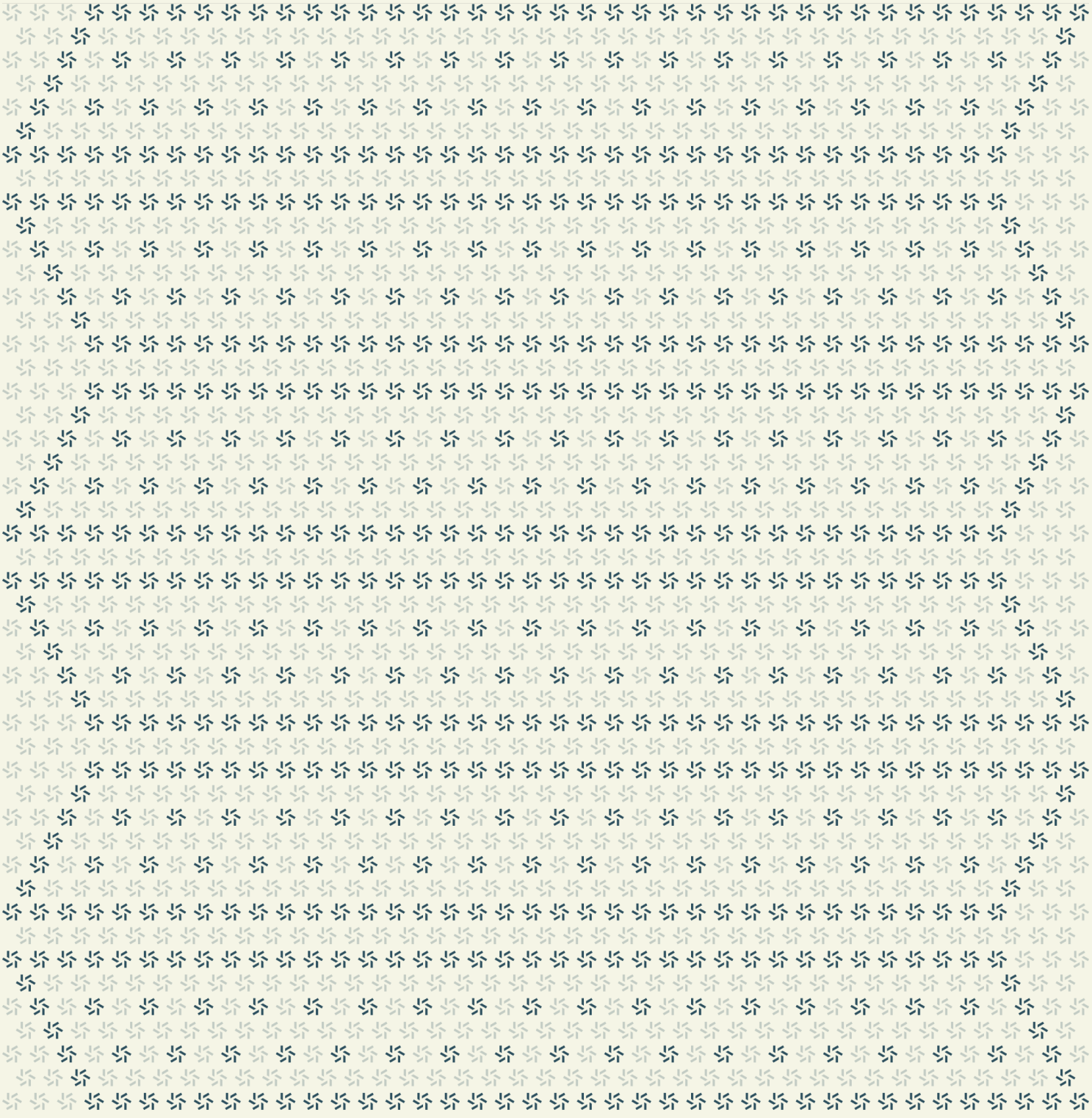
Prepared for
Roderic Deichler
Ondo Finance Inc.

Prepared by
Frank Bachman
Nathanial Lattimer
Zellic

December 29, 2025

Ondo Global Markets

Solana Application Security Assessment



Contents

About Zellic

Zellic is a vulnerability research firm with deep expertise in blockchain security. We specialize in EVM, Move (Aptos and Sui), and Solana as well as Cairo, NEAR, and Cosmos. We review L1s and L2s, cross-chain protocols, wallets and applied cryptography, zero-knowledge circuits, web applications, and more.

Prior to Zellic, we founded the #1 CTF (competitive hacking) team ↗ worldwide in 2020, 2021, and 2023. Our engineers bring a rich set of skills and backgrounds, including cryptography, web security, mobile security, low-level exploitation, and finance. Our background in traditional information security and competitive hacking has enabled us to consistently discover hidden vulnerabilities and develop novel security research, earning us the reputation as the go-to security firm for teams whose rate of innovation outpaces the existing security landscape.

For more on Zellic's ongoing security research initiatives, check out our website zellic.io ↗ and follow @zellic_io ↗ on Twitter. If you are interested in partnering with Zellic, contact us at hello@zellic.io ↗.

1. Overview

1.1. Executive Summary

Zellic conducted a security assessment for Ondo Finance Inc. from December 12th to December 23rd, 2025. During this engagement, Zellic reviewed Ondo Global Markets's code for security vulnerabilities, design issues, and general weaknesses in security posture.

1.2. Goals of the Assessment

In a security assessment, goals are framed in terms of questions that we wish to answer. These questions are agreed upon through close communication between Zellic and the client. In this assessment, we sought to answer the following questions:

- Could an attacker drain protocol funds or mint unbacked tokens without valid authorization?
- Could an unauthorized user bypass Role-Based Access Control or Whitelist restrictions?
- Could an attacker manipulate protocol logic to evade rate limits or cause a denial of service?

1.3. Non-goals and Limitations

We did not assess the following areas that were outside the scope of this engagement:

- Front-end components
- Infrastructure relating to the project
- Key custody

Due to the time-boxed nature of security assessments in general, there are limitations in the coverage an assessment can provide.

1.4. Results

During our assessment on the scoped Ondo Global Markets programs, we discovered one finding, which was informational in nature.

Additionally, Zellic recorded its notes and observations from the assessment for the benefit of Ondo Finance Inc. in the Discussion section (4. ↗).

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	0
Low	0
Informational	1

2. Introduction

2.1. About Ondo Global Markets

Ondo Finance Inc. contributed the following description of Ondo Global Markets:

> Ondo Global Markets is a platform that tokenizes publicly traded U.S. stocks and ETFs. These programs were built to extend Ondo Global Markets, providing the infrastructure necessary for deploying securities-backed tokens to Solana.

2.2. Methodology

During a security assessment, Zellic works through standard phases of security auditing, including both automated testing and manual review. These processes can vary significantly per engagement, but the majority of the time is spent on a thorough manual review of the entire scope.

Alongside a variety of tools and analyzers used on an as-needed basis, Zellic focuses primarily on the following classes of security and reliability issues:

Basic coding mistakes. Many critical vulnerabilities in the past have been caused by simple, surface-level mistakes that could have easily been caught ahead of time by code review. Depending on the engagement, we may also employ sophisticated analyzers such as model checkers, theorem provers, fuzzers, and so on as necessary. We also perform a cursory review of the code to familiarize ourselves with the programs.

Business logic errors. Business logic is the heart of any smart contract application. We examine the specifications and designs for inconsistencies, flaws, and weaknesses that create opportunities for abuse. For example, these include problems like unrealistic tokenomics or dangerous arbitrage opportunities. To the best of our abilities, time permitting, we also review the contract logic to ensure that the code implements the expected functionality as specified in the platform's design documents.

Integration risks. Several well-known exploits have not been the result of any bug within the contract itself; rather, they are an unintended consequence of the contract's interaction with the broader DeFi ecosystem. Time permitting, we review external interactions and summarize the associated risks: for example, flash loan attacks, oracle price manipulation, MEV/sandwich attacks, and so on.

Code maturity. We look for potential improvements in the codebase in general. We look for violations of industry best practices and guidelines and code quality standards. We also provide suggestions for possible optimizations, such as gas optimization, upgradability weaknesses, centralization risks, and so on.

For each finding, Zellic assigns it an impact rating based on its severity and likelihood. There is no hard-and-fast formula for calculating a finding's impact. Instead, we assign it on a case-by-case basis based on our judgment and experience. Both the severity and likelihood of an issue affect

its impact. For instance, a highly severe issue's impact may be attenuated by a low likelihood. We assign the following impact ratings (ordered by importance): Critical, High, Medium, Low, and Informational.

Zellic organizes its reports such that the most important findings come first in the document, rather than being strictly ordered on impact alone. Thus, we may sometimes emphasize an "Informational" finding higher than a "Low" finding. The key distinction is that although certain findings may have the same impact rating, their *importance* may differ. This varies based on various soft factors, like our clients' threat models, their business needs, and so on. We aim to provide useful and actionable advice to our partners considering their long-term goals, rather than a simple list of security issues at present.

Finally, Zellic provides a list of miscellaneous observations that do not have security impact or are not directly related to the scoped programs itself. These observations — found in the Discussion (4. ↗) section of the document — may include suggestions for improving the codebase, or general recommendations, but do not necessarily convey that we suggest a code change.

2.3. Scope

The engagement involved a review of the following targets:

Ondo Global Markets Programs

Type	Rust
Platform	Solana
Target	gm-solana
Repository	https://github.com/ondoprotocol/gm-solana ↗
Version	`77fb5bdd0d192c3e0849ed331d9c2228b5eaa202`
Programs	`ondo-gm`

2.4. Project Overview

Zellic was contracted to perform a security assessment for a total of 2.4 person-weeks. The assessment was conducted by two consultants over the course of 1.2 calendar weeks.

Contact Information

The following project managers were associated with the engagement:

Jacob Goreski
Engagement Manager
jacob@zellic.io

Chad McDonald
Engagement Manager
chad@zellic.io

Pedro Moura
Engagement Manager
pedro@zellic.io

The following consultants were engaged to conduct the assessment:

Frank Bachman
Engineer
frank@zellic.io

Nathanial Lattimer
Engineer
d0nut@zellic.io

2.5. Project Timeline

The key dates of the engagement are detailed below.

December 16, 2025	Kick-off call
December 12, 2025	Start of primary review period
December 23, 2025	End of primary review period

3. Detailed Findings

3.1. Use of Deprecated `realloc` in account closing logic

Target	OndoGM		
Category	Code Maturity	**Severity**	Informational
Likelihood	N/A	**Impact**	Informational

Description

In `batch_close_attestation_accounts`, the instruction manually implements account closing logic. It currently uses `attestation_info.realloc(0, true)?` to resize the account data, which is deprecated.

Additionally, the lamport transfer logic operates directly on the mutable borrow without snapshotting the destination's starting lamports first. While functionally similar in this specific context, sticking to Anchor's standard implementation avoids potential borrow checker edge cases.

Impact

This is primarily a code maturity and best-practice issue. Mirroring Anchor's internal `close` logic would reduce the risk of subtle errors regarding `RefCell` borrows or arithmetic operations during the transfer of lamports.

Recommendations

Update the account closing logic in `batch_close_attestation_accounts` to match Anchor's standard `common::close` implementation.

Recommended Implementation:

```
// Iterate over each attestation account in remaining_accounts
for attestation_info in remaining_accounts.iter() {
    // ... constraints ...

    // Transfer lamports to recipient (Standard Anchor Logic)
    let dest_starting_lamports = self.recipient.lamports();
    **self.recipient.lamports.borrow_mut() = dest_starting_lamports
        .checked_add(attestation_info.lamports())
        .ok_or(OndoError::MathOverflow)?;
    **attestation_info.lamports.borrow_mut() = 0;
```

```
    // Assign account to system program
    attestation_info.assign(&system_program::ID);

    // Resize to zero
    attestation_info.resize(0)?;

    msg!("Attestation account closed: {}", attestation_info.key());
}
```

Remediation

This issue has been acknowledged by Ondo Finance Inc., and fixes were implemented in the following commits:

- 791ee379 ↗
- 64cd6348 ↗

4. Discussion

The purpose of this section is to document miscellaneous observations that we made during the assessment. These discussion notes are not necessarily security related and do not convey that we are suggesting a code change.

4.1. Stale Quote Arbitrage

The protocol validates off-chain attestations using a fixed time window (`MAX_ATTESTATION_EXPIRATION`), currently set to 30 seconds. During this window, an attestation may remain valid even if market conditions move materially.

In periods of elevated volatility, time-windowed attestations can create stale-quote risk, where execution may occur against a price that is no longer representative of prevailing market conditions. This risk is most relevant when rapid price movements occur within the validity window.

On-chain mitigations include `OracleSanityCheck` (which references an admin-controlled price) and `TokenLimit` caps that constrain volume. In addition, the system operates with permissioned access and active monitoring, enabling detection and enforcement against abusive trading behavior, including off-chain controls where appropriate.

This is an assumed risk for the Ondo protocol.

4.2. Malleable Attestation Signatures

Attestations are used to relay a price and quantity of an asset that Ondo Finance Inc. is willing to trade within a given expiration window. These attestations are signed with an secp256k1 ECDSA signature which provides integrity of the attestation against malicious manipulation. The verification process of this signature is done with the secp256k1 solana sysvar program.

An important aspect to understand about ECDSA signatures is that they have limited malleability due to the nature of elliptic curve cryptography. In practice, this means that for a given payload and key, there are two signatures, which can be computed from each other, which correctly verify against the payload and key pair. The typical mitigation is to check that the signature is given in the "low" form and rejecting "high" form signatures (or vice versa). The Solana secp256k1 program does _not_ do this check by default.

A malleable ECDSA signature in itself does not necessarily constitute a vulnerability, but coupled with a design or architecture that depends on independent signatures, could. Thankfully, in the case of Ondo Finance Inc., this behavior does not pose a problem as attestation idempotency is enforced via an "Attestation ID", used in a Program Derived Address, that is also part of the attestation payload. Regardless of the signature order, this attestation ID does not change, meaning that attestations cannot be reused, which would be the typical threat posed here.

4.3. Role Documentation

The program uses a wide variety of roles in its PDA-based RBAC implementation. While this is excellent in providing fine-grained control over the entities able to perform particular actions, and limiting the blast radius of potential compromise, the more complicated an authorization scheme is the more difficult it is to administer and augment.

Generally, we believe Ondo Finance Inc. to have excellent documentation around how their protocol works as well as the purpose of the various roles, but we did provide a suggestion around providing contextual documentation comments in the `RoleType` definition.

As an example:

```
#[derive(Clone, Copy, PartialEq, Eq, AnchorDeserialize, AnchorSerialize)]
pub enum RoleType {
    /// Represents the right to mint USDon tokens directly.
    MinterRoleUSDon,
    /// Represents the right to burn USDon tokens directly.
    BurnerRoleUSDon,
    /// <some explanation that defines the general responsibilities of admin
    and how it differs from the manager counterpart>
    AdminRoleUSDon,
    ...
}
```

This would have the benefit of providing much desired in-context documentation on the intended purpose of a role and whether or not it might be the right fit for a new endpoint or responsibility.

5. Threat Model

As time permitted, we analyzed each instruction in the program and created a written threat model for the most critical instructions. A threat model documents the high-level functionality of a given instruction, the inputs it receives, and the accounts it operates on as well as the main checks performed on them; it gives an overview of the attack surface of the programs and of the level of control an attacker has over the inputs of critical instructions.

For brevity, system accounts and well-known program accounts may not have been included in the list of accounts received by an instruction; the instructions that receive these accounts make use of Anchor types, which automatically ensure that the public key of the account is correct.

Discriminant checks, ownership checks, and rent checks are not discussed for each individual account; unless otherwise stated, the program uses Anchor types, which perform the necessary checks automatically.

Not all instructions in the audit scope may have been modeled. The absence of a threat model in this section does not necessarily suggest that an instruction is safe.

5.1. Program: `ongo-gm`

Instruction: `initialize_usdon_manager`

Initialize the USDon Manager state account. This sets up the manager with the USDon mint, initial price settings, oracle configuration, and vault addresses for USDC and USDon tokens.

Input parameters

- `oracle_price_enabled: bool`: Whether to enable oracle price feeds.
- `oracle_price_max_age: u64`: The maximum age in seconds for oracle price data.
- `usdc_price_update_address: Pubkey`: The address of the USDC price oracle.

Accounts

- **`payer`**: Pays for account creation.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`authority`**: The account with the authority to initialize the USDon Manager.
  - Signer: Yes
  - Init: No
  - PDA: No

    - Writable: No
    - Constraints: N/A
- **`mint_authority`**: The mint authority PDA.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["mint_authority"]`.
- **`usdon_mint`**: The USDon mint account.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: Authority must be `mint_authority`, token program must be `token_2022`.
- **`usdon_vault`**: The USDon vault token account.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: Associated Token Account for `usdon_mint` owned by `usdon_manager_state`. Token program `token_2022`.
- **`usdc_vault`**: The USDC vault token account.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: Associated Token Account for USDC mint owned by `usdon_manager_state`. (On mainnet, mint must be `USDC_MINT`).
- **`usdon_manager_state`**: The USDonManagerState account to be initialized.
    - Signer: No
    - Init: Yes
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `["usdon_manager"]`. Space `8 + USDonManagerState::INIT_SPACE`. Payer `payer`.
- **`authority_role_account`**: The Roles account verifying the authority.
    - Signer: No

- Init: No
- PDA: Yes
- Writable: No
- Constraints: Seeds `["GuardianUSDon", authority.key]`. Verifies authority has `GUARDIAN_USDON` role.

- **`system_program`**: The system program.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A

Additional checks and behavior

- Validates `oracle_price_max_age` is > 0 and <= `MAX_AGE_UPPER_BOUND`.
- Validates `usdc_price_update_address` is not default.

CPI

N/A

Instruction: `initialize_gmtoken_manager`

Initialize the `GmTokenManagerState` account. Sets up global pause states and the attestation signer configuration.

Input parameters

- `factory_paused: bool`: Initial state of the token factory pause flag.
- `redemptions_paused: bool`: Initial state of the global redemption pause flag.
- `minting_paused: bool`: Initial state of the global minting pause flag.
- `attestation_signer_secp: [u8; 20]`: The Ethereum address used for secp256k1 signature verification.
- `trading_hours_offset: i64`: The trading hours offset from UTC in seconds.

Accounts

- **`payer`**: Pays for account creation.

    - Signer: Yes
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: N/A
- **`authority`**: The account with the authority to initialize the GM Token Manager.
    - Signer: Yes
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`gmtoken_manager_state`**: The GmTokenManagerState account to be initialized.
    - Signer: No
    - Init: Yes
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `["gmtoken_manager"]`. Space `8 + GMTokenManagerState::INIT_SPACE`. Payer `payer`.
- **`authority_role_account`**: The Roles account verifying the authority.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["AdminRoleGMTokenManager", authority.key]`. Verifies authority has `ADMIN_ROLE_GMTOKEN_MANAGER` role.
- **`system_program`**: The system program.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A

Additional checks and behavior

- Validates `trading_hours_offset` is within allowed range.

CPI

N/A

Instruction: `set_trading_hours_offset`

Set trading hours offset for the GM token manager.

Input parameters

- `new_trading_hours_offset: i64`: The new timezone offset in seconds from UTC.

Accounts

- **`authority`**: The account with the authority to set the offset.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: The Roles account verifying the authority.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `[role_seed, authority.key]`. Role must be `AdminRoleGMTokenManager` OR `IssuanceHoursRole`.
- **`gmtoken_manager_state`**: The GmTokenManagerState account to be modified.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["gmtoken_manager"]`.

Additional checks and behavior

- Validates `new_trading_hours_offset` is within allowed range.

CPI

N/A

Instruction: `enable_oracle_price`

Enable or disable oracle price for USDon operations.

Input parameters

- `is_enabled: bool`: Whether to enable oracle price feeds.

Accounts

- **`authority`**: The account with the authority to execute the operation.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`usdon_manager_state`**: The USDonManagerState account to be modified.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["usdon_manager"]`.
- **`authority_role_account`**: The Roles account verifying the authority.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["AdminRoleUSDonManager", authority.key]`. Verifies `ADMIN_ROLE_USDON_MANAGER`.

Additional checks and behavior

N/A

CPI

N/A

Instruction: `set_oracle_price_max_age`

Set the maximum age for oracle price data.

Input parameters

- `oracle_price_max_age: u64`: The new maximum age in seconds.

Accounts

- **`authority`**: The account with the authority to execute the operation.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`usdon_manager_state`**: The USDonManagerState account to be modified.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["usdon_manager"]`.
- **`authority_role_account`**: The Roles account verifying the authority.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["AdminRoleUSDonManager", authority.key]`. Verifies `ADMIN_ROLE_USDON_MANAGER`.

Additional checks and behavior

- Validates `oracle_price_max_age` is > 0 and <= `MAX_AGE_UPPER_BOUND`.

CPI

N/A

Instruction: `set_usdc_price_update_address`

Set the USDC price oracle address.

Input parameters

- `new_price_update_address: Pubkey`: The new USDC price oracle public key.

Accounts

- **`authority`**: The account with the authority to execute the operation.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`usdon_manager_state`**: The USDonManagerState account to be modified.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["usdon_manager"]`.
- **`authority_role_account`**: The Roles account verifying the authority.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["AdminRoleUSDonManager", authority.key]`. Verifies `ADMIN_ROLE_USDON_MANAGER`.

Additional checks and behavior

- Validates address is not default.

CPI

N/A

Instruction: `initialize_user`

Initialize a new `OndoUser` account for a user and mint pair.

Input parameters

- `rate_limit: Option<u64>`: Optional rate limit cap.
- `limit_window: Option<u64>`: Optional rate limit window in seconds.

Accounts

- **`payer`**: Pays for account creation.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`authority`**: The account authorizing the initialization.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`user`**: The user for whom the account is being initialized.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`mint`**: The GM Token mint.
  - Signer: No
  - Init: No
  - PDA: No

    - Writable: No
    - Constraints: Interface Account (Mint).
- **`authority_role_account`**: The Roles account verifying the authority.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["AdminRoleGMTokenManager", authority.key]`. Verifies `ADMIN_ROLE_GMTOKEN_MANAGER`.
- **`ondo_user`**: The OndoUser account to be initialized.
    - Signer: No
    - Init: Yes
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `["OndoUser", user.key, mint.key]`. Space `8 + OndoUser::INIT_SPACE`. Payer `payer`.
- **`system_program`**: The system program.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A

Additional checks and behavior

- Initializes capacity used to 0 if rate limits are set.

CPI

N/A

Instruction: `initialize_token_limit`

Initialize a `TokenLimit` account for a GM Token/USDon.

Input parameters

- `rate_limit: Option<u64>`: Global rate limit cap.
- `limit_window: Option<u64>`: Global rate limit window.
- `default_user_rate_limit: Option<u64>`: Default per-user cap.
- `default_limit_window: Option<u64>`: Default per-user window.

Accounts

- **`payer`**: Pays for account creation.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`authority`**: The authority executing the instruction.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`mint`**: The GM Token or USDon mint.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: Interface Account (Mint).
- **`token_limit`**: The TokenLimit account to be initialized.
  - Signer: No
  - Init: Yes
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["token", mint.key]`. Space `8 + TokenLimit::INIT_SPACE`. Payer `payer`.
- **`authority_role_account`**: The Roles account verifying the authority.
  - Signer: No
  - Init: No
  - PDA: Yes

    - Writable: No
    - Constraints: Seeds `["DeployerRoleGMTokenFactory", authority.key]`. Verifies `DEPLOYER_ROLE_GMTOKEN_FACTORY`.
- **`system_program`**: The system program.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A

Additional checks and behavior

- Validates windows are > 0 if provided.

CPI

N/A

Instruction: `set_token_limit`

Set or update the token limit parameters for a GM Token/USDon.

Input parameters

- `rate_limit: Option<u64>`: New global rate limit cap.
- `limit_window: Option<u64>`: New global rate limit window.
- `default_user_rate_limit: Option<u64>`: New default per-user cap.
- `default_user_limit_window: Option<u64>`: New default per-user window.

Accounts

- **`authority`**: The authority executing the instruction.
    - Signer: Yes
    - Init: No
    - PDA: No
    - Writable: Yes (implied for payer of rent logic, though here acts as signer authority)

    - Constraints: N/A
- **`mint`**: The GM Token or USDon mint.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: Interface Account (Mint).
- **`token_limit`**: The TokenLimit account to be updated.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `["token", mint.key]`.
- **`authority_role_account`**: The Roles account verifying the authority.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["AdminRoleGMTokenManager", authority.key]`. Verifies `ADMIN_ROLE_GMTOKEN_MANAGER`.

Additional checks and behavior

- Updates only fields provided as `Some`.
- Initializes usage counters if limits are newly enabled.

CPI

N/A

Instruction: `initialize_sanity_check`

Initialize an OracleSanityCheck state account for a given mint.

Input parameters

- `last_price: u64`: Initial reference price.

- `allowed_deviation_bps: u64`: Allowed deviation in basis points.
- `max_time_delay: i64`: Maximum price age in seconds.

Accounts

- **`payer`**: Pays for account creation.
    - Signer: Yes
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: N/A
- **`authority`**: The authority executing the instruction.
    - Signer: Yes
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`sanity_check`**: The OracleSanityCheck account to be initialized.
    - Signer: No
    - Init: Yes
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `["sanity_check", mint.key]`. Space `8 + OracleSanityCheck::INIT_SPACE`. Payer `payer`.
- **`authority_role_account`**: The Roles account verifying the authority.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["AdminRoleOndoSanityCheck", authority.key]`. Verifies `ADMIN_ROLE_ONDO_SANITY_CHECK`.
- **`mint`**: The GM Token mint.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: Interface Account (Mint).
- **`system_program`**: The system program.

- Signer: No
- Init: No
- PDA: No
- Writable: No
- Constraints: N/A

Additional checks and behavior

- Validates deviation <= 100%.
- Validates time delay <= max expiration.
- Validates price > 0.

CPI

N/A

Instruction: `mint_with_usdon`

Mints GM tokens to the user by accepting USDon from the user. Validates an attestation.

Input parameters

- `attestation_id: [u8; 16]`: Unique ID for the attestation.
- `price: u64`: Price in the attestation.
- `amount: u64`: Amount of GM tokens to mint.
- `expiration: i64`: Expiration timestamp.

Accounts

- **`user`**: The user performing the swap.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`mint`**: The GM Token mint involved in the swap.
  - Signer: No

    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: Authority is `mint_authority`. Token program `token_program`.
- **`mint_authority`**: The mint authority PDA.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["mint_authority"]`.
- **`ondo_user`**: The OndoUser account.
    - Signer: No
    - Init: Yes (if needed)
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `["OndoUser", user.key, mint.key]`. Space `8 + OndoUser::INIT_SPACE`. Payer `user`.
- **`token_limit_account`**: The TokenLimit account.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `["token", mint.key]`.
- **`sanity_check_account`**: The OracleSanityCheck account.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `["sanity_check", mint.key]`.
- **`user_token_account`**: The user's associated token account for the GM Token.
    - Signer: No
    - Init: Yes (if needed)
    - PDA: No
    - Writable: Yes
    - Constraints: Associated Token Account for `mint` owned by `user`. Token program `token_program`.
- **`attestation_id_account`**: The attestation ID account (anti-replay).
    - Signer: No

    - Init: No
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `["attestation_id", attestation_id]`.
- **`whitelist`**: The Whitelist account.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["whitelist", user.key]`.
- **`token_program`**: The Token-2022 program.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`system_program`**: The system program.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`associated_token_program`**: The Associated Token program.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`usdon_vault`**: The USDon vault.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: Owned by `usdon_manager_state`, mint is `usdon_mint`.
- **`usdon_mint`**: The USDon mint.
    - Signer: No
    - Init: No
    - PDA: No

    - Writable: Yes
    - Constraints: Equals `usdon_manager_state.usdon_mint`.
- **`user_usdon_token_account`**: The user's USDon token account.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: N/A
- **`usdon_manager_state`**: The USDonManagerState account.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["usdon_manager"]`.
- **`gmtoken_manager_state`**: The GmTokenManagerState account.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `["gmtoken_manager"]`.
- **`instructions`**: The sysvar instructions account.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: Address `SysvarInstructions`.

Additional checks and behavior

- Verifies secp256k1 signature.
- Validates attestation expiration.
- Validates user whitelist.
- Checks trading hours.
- Performs price sanity check.
- Checks and updates rate limits.
- Transfers USDon from user to vault.
- Mints GM Tokens to user.

CPI

- `token::transfer_checked`: Transfers USDon from user to vault.
- `token::mint_to`: Mints GM tokens to user.

Instruction: `mint_with_usdc`

Mints GM tokens to the user by accepting USDC. Swaps USDC to USDon internally, then burns USDon (conceptually), finally minting GM.

Input parameters

- `attestation_id: [u8; 16]`
- `price: u64`
- `amount: u64`
- `expiration: i64`

Accounts

Includes all accounts from **`mint_with_usdon`** *context (named* **`USDCSwapContext`***), plus:*

- **`spl_token_program`**: The legacy SPL Token program.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`usdc_price_update`**: The USDC oracle account.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: Checked via `usdon_manager_state.usdc_price_update`.
- **`usdc_vault`**: The USDC vault.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: Yes

    - Constraints: Owned by `usdon_manager_state`, mint is `usdc_mint`.
- **`usdc_mint`**: The USDC mint.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: Matches `USDC_MINT`.
- **`user_usdc_token_account`**: The user's USDC token account.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: N/A

Additional checks and behavior

- Validates USDC oracle price (if enabled).
- Transfers USDC from user to vault.
- Burns USDon from vault (after virtual swap).
- Mints GM tokens to user.
- All other checks (signature, whitelist, hours, limits) apply.

CPI

- `spl_token::transfer_checked`: USDC User -> Vault.
- `token_2022::burn_checked`: USDon Vault -> Burn.
- `token_2022::mint_to`: GM Token -> User.

Instruction: `redeem_for_usdon`

Redeems GM tokens for USDon.

Input parameters

- `attestation_id: [u8; 16]`
- `price: u64`
- `amount: u64`

- `expiration: i64`

Accounts

Uses **`USDonSwapContext`**. *See* **`mint_with_usdon`**.

Additional checks and behavior

- Verifies secp256k1 signature (Sell side).
- Checks pauses (redemption).
- Burns GM tokens from user.
- Mints USDon to user.

CPI

- `token_2022::mint_to`: USDon Mint -> User.
- `token_2022::burn_checked`: GM Token User -> Burn.

Instruction: `redeem_for_usdc`

Redeems GM tokens for USDC.

Input parameters

- `attestation_id: [u8; 16]`
- `price: u64`
- `amount: u64`
- `expiration: i64`

Accounts

Uses **`USDCSwapContext`**. *See* **`mint_with_usdc`**.

Additional checks and behavior

- Burns GM tokens from user.
- Mints USDon (intermediate).

- Swaps USDon to USDC: Transfers USDon User->Vault, Transfers USDC Vault->User.

CPI

- `token_2022::mint_to`: USDon -> User.
- `token_2022::transfer_checked`: USDon User -> Vault.
- `spl_token::transfer_checked`: USDC Vault -> User.
- `token_2022::burn_checked`: GM Token User -> Burn.

Instruction: `add_to_whitelist`

Add an address to the whitelist.

Input parameters

- `address_to_whitelist: Pubkey`: The address to whitelist.

Accounts

- **`payer`**: Pays for account creation.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`authority`**: The authority executing the instruction.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: The Roles account verifying the authority.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No

    - Constraints: Seeds `["AdminRoleWhitelist", authority.key]`. Verifies `ADMIN_ROLE_WHITELIST`.
- **`whitelist`**: The whitelist marker account.
    - Signer: No
    - Init: Yes
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `["whitelist", address_to_whitelist]`. Space `8`. Payer `payer`.
- **`system_program`**: The system program.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A

Additional checks and behavior

N/A

CPI

N/A

Instruction: `remove_from_whitelist`

Remove an address from the whitelist.

Input parameters

- `address_to_remove: Pubkey`: The address to remove.

Accounts

- **`authority`**: The authority executing the instruction.
    - Signer: Yes
    - Init: No

    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`recipient`**: Receives rent from closed account.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: N/A
- **`authority_role_account`**: The Roles account verifying the authority.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["AdminRoleWhitelist", authority.key]`. Verifies `ADMIN_ROLE_WHITELIST`.
- **`whitelist`**: The whitelist account to close.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `["whitelist", address_to_remove]`. Close to `recipient`.

Additional checks and behavior

N/A

CPI

N/A

Instruction: `grant_role`

Grants a specified role to a user. Signer must be the program upgrade authority.

Input parameters

- `role: RoleType`: The role to grant.
- `user: Pubkey`: The user address.

Accounts

- **`payer`**: Pays for account creation.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`authority`**: The program upgrade authority.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`role_to_grant`**: The Roles account to create.
  - Signer: No
  - Init: Yes
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `[role.seed(), user]`. Space `Roles::INIT_SPACE`. Payer `payer`.
- **`system_program`**: The system program.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`program`**: The Ondo GM program address.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: Address matches program ID.

- **`program_data`**: The ProgramData account.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: `upgrade_authority_address` == `authority.key`.

Additional checks and behavior

- Verifies `program_data` address via `program.programdata_address()`.

CPI

N/A

Instruction: `grant_usdon_role`

Grants a USDon role (`MinterRoleUsdon`, `BurnerRoleUsdon`) to a user.

Input parameters

- `role: RoleType`: The role to grant.
- `user: Pubkey`: The user address.

Accounts

- **`payer`**: Pays for account creation.
    - Signer: Yes
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: N/A
- **`authority`**: The authority executing the instruction.
    - Signer: Yes
    - Init: No
    - PDA: No

- Writable: No
- Constraints: N/A

- **`authority_role_account`**: The Roles account verifying the authority.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["GuardianUSDon", authority.key]`. Verifies `GUARDIAN_USDON`.
- **`role_to_grant`**: The Roles account to create.
  - Signer: No
  - Init: Yes
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `[role.seed(), user]`. Space `Roles::INIT_SPACE`. Payer `payer`.
- **`system_program`**: The system program.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A

Additional checks and behavior

- Validates role is `MinterRoleUsdon` or `BurnerRoleUsdon`.

CPI

N/A

Instruction: `revoke_usdon_role`

Revokes a USDon role from a user.

Input parameters

N/A

Accounts

- **`authority`**: The authority executing the instruction.
    - Signer: Yes
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`recipient`**: Receives rent.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: N/A
- **`authority_role_account`**: The Roles account verifying the authority.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["GuardianUSDon", authority.key]`. Verifies `GUARDIAN_USDON`.
- **`role_to_revoke`**: The Roles account to close.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `[role.seed(), user]`. Close to `recipient`.

Additional checks and behavior

- Validates role being revoked is `MinterRoleUsdon` or `BurnerRoleUsdon`.

CPI

N/A

Instruction: `mint_usdon`

Mint USDon tokens (Admin function).

Input parameters

- `amount: u64`: Amount to mint.

Accounts

- **`authority`**: The authority executing the instruction.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`mint_authority`**: The mint authority PDA.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["mint_authority"]`.
- **`usdon_manager_state`**: The USDon state account.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["usdon_manager"]`.
- **`authority_role_account`**: The Roles account verifying the authority.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No

    - Constraints: Verifies `MinterRoleUSDon` OR `AdminRoleUSDon`.
- **`mint`**: The USDon mint.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: Address matches `usdon_manager_state.usdon_mint`. Authority `mint_authority`.
- **`token_program`**: The Token-2022 program.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`destination`**: The destination token account.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: N/A

Additional checks and behavior

- `amount` > 0 and <= `MAX_MINT_AMOUNT`.

CPI

- `token_2022::mint_to`: Mints tokens.

Instruction: `burn_usdon`

Burn USDon tokens (Admin function).

Input parameters

- `amount: u64`: Amount to burn.

Accounts

- **`authority`**: The authority executing the instruction.
    - Signer: Yes
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: N/A
- **`permanent_delegate`**: The permanent delegate PDA (Mint Authority).
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["mint_authority"]`.
- **`usdon_manager_state`**: The USDon state account.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["usdon_manager"]`.
- **`authority_role_account`**: The Roles account verifying the authority.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Verifies `BurnerRoleUSDon` OR `AdminRoleUSDon`.
- **`mint`**: The USDon mint.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: Address matches `usdon_manager_state.usdon_mint`. Authority `permanent_delegate`.
- **`token_program`**: The Token-2022 program.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No

    - Constraints: N/A
- **`destination`**: The token account to burn from.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: Mint matches `mint`.

Additional checks and behavior

- `amount` > 0.

CPI

- `token_2022::burn_checked`: Burns tokens using delegate authority.

Instruction: `mint_gm`

Mint GM Tokens (Admin function).

Input parameters

- `amount: u64`: Amount to mint.

Accounts

- **`payer`**: Pays for init if needed.
    - Signer: Yes
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: N/A
- **`authority`**: The authority executing the instruction.
    - Signer: Yes
    - Init: No
    - PDA: No

    - Writable: No
    - Constraints: N/A
- **`user`**: The user receiving tokens.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: Unchecked (Associated Token checks used).
- **`authority_role_account`**: The Roles account verifying the authority.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["MinterRoleGMToken", authority.key]`. Verifies `MINTER_ROLE_GMTOKEN`.
- **`oracle_sanity_check`**: The sanity check account.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `["sanity_check", mint.key]`. Matches `mint`.
- **`mint_authority`**: The mint authority PDA.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["mint_authority"]`.
- **`mint`**: The GM Token mint.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: Authority `mint_authority`. Not `usdon_mint`.
- **`destination`**: The destination token account.
    - Signer: No
    - Init: Yes (if needed)
    - PDA: No
    - Writable: Yes

    - Constraints: Associated token account for `mint` and `user`.
- **`usdon_manager_state`**: The USDon state (for mint check).
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["usdon_manager"]`.
- **`token_program`**: The Token-2022 program.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`associated_token_program`**: The Associated Token program.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`system_program`**: The system program.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A

Additional checks and behavior

- Checks amount > 0.
- Calculates notional USD value using sanity check price.
- Checks notional value <= `MAX_MINT_AMOUNT`.

CPI

- `token_2022::mint_to`: Mints tokens.

Instruction: `grant_gmtoken_role`

Grant a GM Token role (`Minter`, `Pauser`, `Unpauser`).

Input parameters

- `role: RoleType`: Role to grant.
- `user: Pubkey`: User address.

Accounts

- **`payer`**: Pays for account.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`authority`**: Authority executing.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `ADMIN_ROLE_GMTOKEN`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["AdminRoleGMToken", authority.key]`.
- **`role_to_grant`**: Account to create.
  - Signer: No
  - Init: Yes
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `[role.seed(), user]`. Space `Roles::INIT_SPACE`. Payer `payer`.
- **`system_program`**: System program.
  - Signer: No

- Init: No
- PDA: No
- Writable: No
- Constraints: N/A

Additional checks and behavior

- Validates role is `MinterRoleGMToken`, `PauserRoleGMToken`, or `UnpauserRoleGMToken`.

CPI

N/A

Instruction: `revoke_gmtoken_role`

Revoke a GM Token role.

Input parameters

N/A

Accounts

- **`recipient`**: Receives rent.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`authority`**: Authority executing.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `ADMIN_ROLE_GMTOKEN`.

- Signer: No
- Init: No
- PDA: Yes
- Writable: No
- Constraints: Seeds `["AdminRoleGMToken", authority.key]`.

- **`role_to_revoke`**: Account to close.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `[role.seed(), user]`. Close to `recipient`.

Additional checks and behavior

- Validates role matches allowed GM Token roles.

CPI

N/A

Instruction: `grant_gmtoken_factory_role`

Grant a Factory role (`Pauser`, `Deployer`).

Input parameters

- `role: RoleType`: Role to grant.
- `user: Pubkey`: User address.

Accounts

- **`payer`**: Pays for account.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A

- **`authority`**: Authority executing.
    - Signer: Yes
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`authority_role_account`**: Verifies `ADMIN_ROLE_GMTOKEN_FACTORY`.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["AdminRoleGMTokenFactory", authority.key]`.
- **`role_to_grant`**: Account to create.
    - Signer: No
    - Init: Yes
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `[role.seed(), user]`. Space `Roles::INIT_SPACE`. Payer `payer`.
- **`system_program`**: System program.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A

Additional checks and behavior

- Validates role is `PauserRoleGMTokenFactory` or `DeployerRoleGMTokenFactory`.

CPI

N/A

Instruction: `revoke_gmtoken_factory_role`

Revoke a Factory role.

Input parameters

N/A

Accounts

- **`authority`**: Authority executing.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`recipient`**: Receives rent.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`authority_role_account`**: Verifies `ADMIN_ROLE_GMTOKEN_FACTORY`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["AdminRoleGMTokenFactory", authority.key]`.
- **`role_to_revoke`**: Account to close.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `[role.seed(), user]`. Close to `recipient`.

Additional checks and behavior

- Validates role matches allowed Factory roles.

CPI

N/A

Instruction: `pause_token_factory`

Pause the GM Token Factory.

Input parameters

N/A

Accounts

- **`authority`**: Authority executing.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `PAUSER_ROLE_GMTOKEN_FACTORY`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["PauserRoleGMTokenFactory", authority.key]`.
- **`gmtoken_manager_state`**: State account.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["gmtoken_manager"]`.

Additional checks and behavior

- Sets `factory_paused` = true.

CPI

N/A

Instruction: `pause_token_factory_admin`

Pause the GM Token Factory (Admin).

Input parameters

N/A

Accounts

- **`authority`**: Authority executing.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `ADMIN_ROLE_GMTOKEN_FACTORY`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["AdminRoleGMTokenFactory", authority.key]`.
- **`gmtoken_manager_state`**: State account.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["gmtoken_manager"]`.

Additional checks and behavior

- Sets `factory_paused` = true.

CPI

N/A

Instruction: `resume_token_factory`

Resume the GM Token Factory.

Input parameters

N/A

Accounts

- **`authority`**: Authority executing.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `ADMIN_ROLE_GMTOKEN_FACTORY`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["AdminRoleGMTokenFactory", authority.key]`.
- **`gmtoken_manager_state`**: State account.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["gmtoken_manager"]`.

Additional checks and behavior

- Sets `factory_paused` = false.

CPI

N/A

Instruction: `pause_token`

Pause a specific GM Token (SPL extension).

Input parameters

N/A

Accounts

- **`authority`**: Authority executing.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`authority_role_account`**: Verifies `PAUSER_ROLE_GMTOKEN`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["PauserRoleGMToken", authority.key]`.
- **`mint`**: The token mint.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: Authority `mint_authority`. Not `usdon_mint`.
- **`mint_authority`**: PDA.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["mint_authority"]`.
- **`usdon_manager_state`**: State.
  - Signer: No

    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["usdon_manager"]`.
- **`token_program`**: Token-2022.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A

Additional checks and behavior

N/A

CPI

- `spl_token_2022::instruction::pause`: Sets mint to paused state.

Instruction: `resume_token`

Resume a specific GM Token (SPL extension).

Input parameters

N/A

Accounts

- **`authority`**: Authority executing.
    - Signer: Yes
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: N/A
- **`authority_role_account`**: Verifies `UNPAUSER_ROLE_GMTOKEN`.

    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["UnpauserRoleGMToken", authority.key]`.
- **`mint`**: The token mint.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: Authority `mint_authority`. Not `usdon_mint`.
- **`mint_authority`**: PDA.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["mint_authority"]`.
- **`usdon_manager_state`**: State.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["usdon_manager"]`.
- **`token_program`**: Token-2022.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A

Additional checks and behavior

N/A

CPI

- `spl_token_2022::instruction::resume`: Sets mint to unpaused state.

Instruction: `pause_global_minting`

Pause global minting flag (Permissioned).

Input parameters

N/A

Accounts

- **`authority`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `PAUSER_ROLE_GMTOKEN_MANAGER`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["PauserRoleGMTokenManager", authority.key]`.
- **`gmtoken_manager_state`**: State.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["gmtoken_manager"]`.

Additional checks and behavior

- Sets `minting_paused` = true.

CPI

N/A

Instruction: `resume_global_minting`

Resume global minting flag (Admin).

Input parameters

N/A

Accounts

- **`authority`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `ADMIN_ROLE_GMTOKEN_MANAGER`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["AdminRoleGMTokenManager", authority.key]`.
- **`gmtoken_manager_state`**: State.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["gmtoken_manager"]`.

Additional checks and behavior

- Sets `minting_paused` = false.

CPI

N/A

Instruction: `pause_global_minting_admin`

Pause global minting flag (Admin).

Input parameters

N/A

Accounts

Same as **`resume_global_minting`**.

Additional checks and behavior

- Sets `minting_paused` = true.

CPI

N/A

Instruction: `pause_global_redemption`

Pause global redemption flag (Permissioned).

Input parameters

N/A

Accounts

- **`authority`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No

    - Constraints: N/A
- **`authority_role_account`**: Verifies `PAUSER_ROLE_GMTOKEN_MANAGER`.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["PauserRoleGMTokenManager", authority.key]`.
- **`gmtoken_manager_state`**: State.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `["gmtoken_manager"]`.

Additional checks and behavior

- Sets `redemption_paused` = true.

CPI

N/A

Instruction: `resume_global_redemption`

Resume global redemption flag (Admin).

Input parameters

N/A

Accounts

- **`authority`**: Signer.
    - Signer: Yes
    - Init: No
    - PDA: No

    - Writable: No
    - Constraints: N/A
- **`authority_role_account`**: Verifies `ADMIN_ROLE_GMTOKEN_MANAGER`.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["AdminRoleGMTokenManager", authority.key]`.
- **`gmtoken_manager_state`**: State.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `["gmtoken_manager"]`.

Additional checks and behavior

- Sets `redemption_paused` = false.

CPI

N/A

Instruction: `pause_global_redemption_admin`

Pause global redemption flag (Admin).

Input parameters

N/A

Accounts

Same as **`resume_global_redemption`**.

Additional checks and behavior

- Sets `redemption_paused` = true.

CPI

N/A

Instruction: `pause_token_redemption`

Pause redemption for a specific token (Permissioned).

Input parameters

N/A

Accounts

- **`authority`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `PAUSER_ROLE_GMTOKEN_MANAGER`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["PauserRoleGMTokenManager", authority.key]`.
- **`token_limit_account`**: Token limits.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["token", mint]`.

Additional checks and behavior

- Sets `redemption_paused` = true on token limit account.

CPI

N/A

Instruction: `resume_token_redemption`

Resume redemption for a specific token (Admin).

Input parameters

N/A

Accounts

- **`authority`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `ADMIN_ROLE_GMTOKEN_MANAGER`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["AdminRoleGMTokenManager", authority.key]`.
- **`token_limit_account`**: Token limits.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["token", mint]`.

Additional checks and behavior

- Sets `redemption_paused` = false.

CPI

N/A

Instruction: `pause_token_redemption_admin`

Pause redemption for a specific token (Admin).

Input parameters

N/A

Accounts

Same as **`resume_token_redemption`**.

Additional checks and behavior

- Sets `redemption_paused` = true.

CPI

N/A

Instruction: `pause_token_minting`

Pause minting for a specific token (Permissioned).

Input parameters

N/A

Accounts

- **`authority`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `PAUSER_ROLE_GMTOKEN_MANAGER`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["PauserRoleGMTokenManager", authority.key]`.
- **`token_limit_account`**: Token limits.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["token", mint]`.

Additional checks and behavior

- Sets `minting_paused` = true.

CPI

N/A

Instruction: `resume_token_minting`

Resume minting for a specific token (Admin).

Input parameters

N/A

Accounts

- **`authority`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `ADMIN_ROLE_GMTOKEN_MANAGER`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["AdminRoleGMTokenManager", authority.key]`.
- **`token_limit_account`**: Token limits.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["token", mint]`.

Additional checks and behavior

- Sets `minting_paused` = false.

CPI

N/A

Instruction: `pause_token_minting_admin`

Pause minting for a specific token (Admin).

Input parameters

N/A

Accounts

Same as **`resume_token_minting`**.

Additional checks and behavior

- Sets `minting_paused` = true.

CPI

N/A

Instruction: `set_ondo_user_limits`

Set user-specific limits.

Input parameters

- `rate_limit: u64`: Rate limit cap.
- `limit_window: u64`: Rate limit window.

Accounts

- **`authority`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`mint`**: The token mint.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: Interface Account (Mint).
- **`authority_role_account`**: Verifies `ADMIN_ROLE_GMTOKEN_MANAGER`.

    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["AdminRoleGMTokenManager", authority.key]`.
- **`ondo_user`**: User state.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `["OndoUser", owner, mint]`.
- **`token_limit`**: Token state.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["token", mint]`. Matches `mint`.

Additional checks and behavior

- Applies default window from `token_limit` if `limit_window` is 0.
- Initializes usage counters.

CPI

N/A

Instruction: `revoke_role`

Revokes a generic role.

Input parameters

- `_role: RoleType`

Accounts

- **`recipient`**: Receives rent.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: N/A
- **`authority`**: Upgrade Authority Signer.
    - Signer: Yes
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`role_to_revoke`**: Account to close.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `[role.seed(), address]`.
- **`system_program`**: System.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`program`**: The program itself.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: Address matches ID.
- **`program_data`**: Program Data.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: `upgrade_authority_address` == `authority`.

Additional checks and behavior

- Verifies upgrade authority matches authority.

CPI

N/A

Instruction: `init_mint`

Initialize GM Token mint (No permanent delegate).

Input parameters

- `name: String`
- `symbol: String`
- `uri: String`
- `freeze_authority: Pubkey`

Accounts

- **`payer`**: Pays for init.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`authority`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `DEPLOYER_ROLE_GMTOKEN_FACTORY`.
  - Signer: No
  - Init: No
  - PDA: Yes

    - Writable: No
    - Constraints: Seeds `["DeployerRoleGMTokenFactory", authority.key]`.
- **`mint`**: The new mint.
    - Signer: Yes
    - Init: No (Manually via CPI)
    - PDA: No
    - Writable: Yes
    - Constraints: N/A
- **`mint_authority`**: PDA.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["mint_authority"]`.
- **`system_program`**: System.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`token_program`**: Token-2022.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`gmtoken_manager_state`**: State.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["gmtoken_manager"]`.

Additional checks and behavior

- Checks factory pause state.
- Initializes extensions: ScaledUiAmount, MetadataPointer, Pausable, ConfidentialTransferMint, TransferHook, DefaultAccountState.

- Initializes Mint and Metadata.

CPI

- Multiple System/Token CPIs to setup mint.

Instruction: `init_mint_delegate`

Initialize USDon mint (With permanent delegate).

Input parameters

- `name: String`
- `symbol: String`
- `uri: String`

Accounts

Same as **`init_mint`**.

Additional checks and behavior

- Adds `PermanentDelegate` extension.

CPI

- Multiple System/Token CPIs to setup mint.

Instruction: `set_attestation_signer_secp`

Set attestation signer address.

Input parameters

- `attestation_signer_secp: [u8; 20]`

Accounts

- **`authority`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `ADMIN_ROLE_GMTOKEN_MANAGER`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["AdminRoleGMTokenManager", authority.key]`.
- **`gmtoken_manager_state`**: State.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["gmtoken_manager"]`.

Additional checks and behavior

N/A

CPI

N/A

Instruction: `grant_gmtoken_manager_role`

Grant GM Token Manager role (`Pauser`, `IssuanceHours`).

Input parameters

- `role: RoleType`
- `user: Pubkey`

Accounts

- **`payer`**: Pays for account.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`authority`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `ADMIN_ROLE_GMTOKEN_MANAGER`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["AdminRoleGMTokenManager", authority.key]`.
- **`role_to_grant`**: Account to create.
  - Signer: No
  - Init: Yes
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `[role.seed(), user]`. Space `Roles::INIT_SPACE`. Payer `payer`.
- **`system_program`**: System.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A

Additional checks and behavior

- Validates role is `PauserRoleGmtokenManager` or `IssuanceHoursRole`.

CPI

N/A

Instruction: `revoke_gmtoken_manager_role`

Revoke GM Token Manager role.

Accounts

- **`authority`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`recipient`**: Rent receiver.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`authority_role_account`**: Verifies `ADMIN_ROLE_GMTOKEN_MANAGER`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["AdminRoleGMTokenManager", authority.key]`.
- **`role_to_revoke`**: Account to close.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `[role.seed(), user]`. Close to `recipient`.

Additional checks and behavior

- Validates role.

CPI

N/A

Instruction: `grant_sanity_setter_role`

Grant `SetterRoleOndoSanityCheck`.

Input parameters

- `role: RoleType`
- `user: Pubkey`

Accounts

- **`payer`**: Pays for account.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`authority`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `ADMIN_ROLE_ONDO_SANITY_CHECK`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["AdminRoleOndoSanityCheck", authority.key]`.

- **`role_to_grant`**: Account to create.
    - Signer: No
    - Init: Yes
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `[role.seed(), user]`. Space `Roles::INIT_SPACE`. Payer `payer`.
- **`system_program`**: System.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A

Additional checks and behavior

- Validates role.

CPI

N/A

Instruction: `revoke_sanity_setter_role`

Revoke `SetterRoleOndoSanityCheck`.

Accounts

- **`recipient`**: Rent receiver.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: N/A
- **`authority`**: Signer.
    - Signer: Yes

    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`authority_role_account`**: Verifies `ADMIN_ROLE_ONDO_SANITY_CHECK`.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["AdminRoleOndoSanityCheck", authority.key]`.
- **`role_to_revoke`**: Account to close.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `[role.seed(), user]`. Close to `recipient`.
- **`system_program`**: System.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A

Additional checks and behavior

- Validates role.

CPI

N/A

Instruction: `set_last_price`

Update sanity check price.

Input parameters

- `last_price: u64`

Accounts

- **`authority`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `SETTER_ROLE_ONDO_SANITY_CHECK`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["SetterRoleOndoSanityCheck", authority.key]`.
- **`mint`**: Token mint.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: Interface Account (Mint).
- **`sanity_check_account`**: State.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["sanity_check", mint]`.

Additional checks and behavior

- `last_price` > 0.

CPI

N/A

Instruction: `grant_sanity_configurer_role`

Grant `ConfigurerRoleOndoSanityCheck`.

Input parameters

- `role: RoleType`
- `user: Pubkey`

Accounts

- **`payer`**: Pays for account.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`authority`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `ADMIN_ROLE_ONDO_SANITY_CHECK`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["AdminRoleOndoSanityCheck", authority.key]`.
- **`role_to_grant`**: Account to create.
  - Signer: No
  - Init: Yes
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `[role.seed(), user]`. Space `Roles::INIT_SPACE`. Payer `payer`.

- **`system_program`**: System.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A

Additional checks and behavior

- Validates role.

CPI

N/A

Instruction: `revoke_sanity_configurer_role`

Revoke `ConfigurerRoleOndoSanityCheck`.

Accounts

- **`recipient`**: Rent receiver.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: N/A
- **`authority`**: Signer.
    - Signer: Yes
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`authority_role_account`**: Verifies `ADMIN_ROLE_ONDO_SANITY_CHECK`.
    - Signer: No
    - Init: No

    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["AdminRoleOndoSanityCheck", authority.key]`.
- **`role_to_revoke`**: Account to close.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: Yes
    - Constraints: Seeds `[role.seed(), user]`. Close to `recipient`.
- **`system_program`**: System.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A

Additional checks and behavior

- Validates role.

CPI

N/A

Instruction: `set_max_time_delay`

Set sanity check time delay.

Input parameters

- `max_time_delay: i64`

Accounts

- **`authority`**: Signer.
    - Signer: Yes

- Init: No
- PDA: No
- Writable: No
- Constraints: N/A

- **`authority_role_account`**: Verifies `CONFIGURER_ROLE_ONDO_SANITY_CHECK`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["ConfigurerRoleOndoSanityCheck", authority.key]`.
- **`mint`**: Token mint.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: Interface Account (Mint).
- **`sanity_check_account`**: State.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["sanity_check", mint]`.

Additional checks and behavior

- Validates `<= MAX_SECONDS_EXPIRATION`.

CPI

N/A

Instruction: `set_allowed_deviation_bps`

Set sanity check deviation.

Input parameters

- `allowed_deviation_bps: u64`

Accounts

Same as **`set_max_time_delay`**.

Additional checks and behavior

- Validates `<= BASIS_POINTS_DIVISOR`.

CPI

N/A

Instruction: `update_scaled_ui_multiplier`

Update UI multiplier via extension.

Input parameters

- `new_multiplier: f64`
- `timestamp: i64`

Accounts

- **`authority`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`authority_role_account`**: Verifies `UPDATE_MULTIPLIER_ROLE`.
  - Signer: No
  - Init: No

    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["UpdateMultiplierRole", authority.key]`.
- **`mint_authority`**: PDA.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["mint_authority"]`.
- **`mint`**: Token mint.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: Authority `mint_authority`. Not `usdon_mint`.
- **`usdon_manager_state`**: State.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["usdon_manager"]`.
- **`token_program`**: Token-2022.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A

Additional checks and behavior

N/A

CPI

- `scaled_ui_amount::update_multiplier`: Updates extension.

Instruction: `update_token_metadata`

Update token metadata.

Input parameters

- `new_name: Option<String>`
- `new_symbol: Option<String>`
- `new_uri: Option<String>`

Accounts

- **`payer`**: Pays shortfall.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`authority`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`authority_role_account`**: Verifies `UPDATE_METADATA_ROLE`.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["UpdateMetadataRole", authority.key]`.
- **`mint_authority`**: PDA.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: No
  - Constraints: Seeds `["mint_authority"]`.
- **`mint`**: Token mint.
  - Signer: No

    - Init: No
    - PDA: No
    - Writable: Yes
    - Constraints: Authority `mint_authority`. Not `usdon_mint`.
- **`usdon_manager_state`**: State.
    - Signer: No
    - Init: No
    - PDA: Yes
    - Writable: No
    - Constraints: Seeds `["usdon_manager"]`.
- **`token_program`**: Token-2022.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A
- **`system_program`**: System.
    - Signer: No
    - Init: No
    - PDA: No
    - Writable: No
    - Constraints: N/A

Additional checks and behavior

- Validates length.
- Calculates rent shortfall and transfers from payer to mint.

CPI

- `token_metadata_interface::update_field`: Updates metadata.
- `system_instruction::transfer`: Funds mint account.

Instruction: `close_attestation_account`

Close an old attestation account.

Input parameters

- `_attestation_id: [u8; 16]`

Accounts

- **`closer`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`attestation`**: Account to close.
  - Signer: No
  - Init: No
  - PDA: Yes
  - Writable: Yes
  - Constraints: Seeds `["attestation_id", id]`. Close to `recipient`.
- **`recipient`**: Creator of attestation.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: Must be `attestation.creator`.
- **`system_program`**: System.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A

Additional checks and behavior

- Validates age > 30s (`MAX_ATTESTATION_EXPIRATION`).

CPI

N/A

Instruction: `batch_close_attestation_accounts`

Batch close attestation accounts.

Input parameters

N/A

Accounts

- **`closer`**: Signer.
  - Signer: Yes
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A
- **`recipient`**: Rent receiver.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: Yes
  - Constraints: N/A
- **`system_program`**: System.
  - Signer: No
  - Init: No
  - PDA: No
  - Writable: No
  - Constraints: N/A

Additional checks and behavior

- Iterates `remaining_accounts`.
- Checks program ownership.
- Checks recipient == creator.
- Checks age.
- Transfers lamports, reallocs to 0, assigns to system program.

CPI

N/A

6. Assessment Results

During our assessment on the scoped Ondo Global Markets programs, we discovered one finding, which was informational in nature.

6.1. Disclaimer

This assessment does not provide any warranties about finding all possible issues within its scope; in other words, the evaluation results do not guarantee the absence of any subsequent issues. Zellic, of course, also cannot make guarantees about any code added to the project after the version reviewed during our assessment. Furthermore, because a single assessment can never be considered comprehensive, we always recommend multiple independent assessments paired with a bug bounty program.

For each finding, Zellic provides a recommended solution. All code samples in these recommendations are intended to convey how an issue may be resolved (i.e., the idea), but they may not be tested or functional code. These recommendations are not exhaustive, and we encourage our partners to consider them as a starting point for further discussion. We are happy to provide additional guidance and advice as needed.

Finally, the contents of this assessment report are for informational purposes only; do not construe any information in this report as legal, tax, investment, or financial advice. Nothing contained in this report constitutes a solicitation or endorsement of a project by Zellic.