Incremental Review (12/17/2022)

0. Scope

```
$ git log -1
commit 4025e1f5a1536036e2719f9bd6b72388d58e0ee0 (HEAD -> vbrunet/2022_11_28-integration-credix, origin/vbrunet/2022_11_28-integration-credix)
Author: Vincent Brunet <vincent.brunet.us@gmail.com>
Date:   Mon Dec 12 19:18:20 2022 +0100

    double-check-credix-pass-fees


$ git diff --stat ae2aad9e e7a26784 programs/uxd/src ':!programs/uxd/src/test/'
 programs/uxd/src/error.rs                                                  |  39 ++++++
 programs/uxd/src/events.rs                                                 |  89 ++++++++++++
 programs/uxd/src/instructions/credix_lp/collect_profit_of_credix_lp_depository.rs | 464 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 programs/uxd/src/instructions/credix_lp/edit_credix_lp_depository.rs      | 127 +++++++++++++++++
 programs/uxd/src/instructions/credix_lp/mint_with_credix_lp_depository.rs | 491 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 programs/uxd/src/instructions/credix_lp/mod.rs                            |  11 ++
 programs/uxd/src/instructions/credix_lp/redeem_from_credix_lp_depository.rs | 499 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 programs/uxd/src/instructions/credix_lp/register_credix_lp_depository.rs  | 180 ++++++++++++++++++++++++
 programs/uxd/src/instructions/edit_controller.rs                           |   3 +
 programs/uxd/src/instructions/edit_identity_depository.rs                  |   9 ++
 programs/uxd/src/instructions/edit_mercurial_vault_depository.rs           |   9 ++
 programs/uxd/src/instructions/freeze_program.rs                            |  40 ++++++
 programs/uxd/src/instructions/initialize_controller.rs                     |   2 +
 programs/uxd/src/instructions/initialize_identity_depository.rs            |   3 +
 programs/uxd/src/instructions/mercurial/mint_with_mercurial_vault_depository.rs |   3 +
 programs/uxd/src/instructions/mercurial/redeem_from_mercurial_vault_depository.rs |   3 +
 programs/uxd/src/instructions/mint_with_identity_depository.rs             |   3 +
 programs/uxd/src/instructions/mod.rs                                       |   4 +
 programs/uxd/src/instructions/redeem_from_identity_depository.rs           |   3 +
 programs/uxd/src/instructions/register_mercurial_vault_depository.rs       |   3 +
 programs/uxd/src/lib.rs                                                    |  93 +++++++++++++
 programs/uxd/src/state/controller.rs                                       |  61 +++++++-
 programs/uxd/src/state/credix_lp_depository.rs                             | 162 ++++++++++++++++++++++
 programs/uxd/src/state/mod.rs                                              |   2 +
 programs/uxd/src/utils/calculate_amount_less_fees.rs                       |  12 ++
 programs/uxd/src/utils/fees.rs                                             |  28 ----
 programs/uxd/src/utils/math/compute_amount_less_fraction.rs                |  20 +++
 programs/uxd/src/utils/math/compute_decrease.rs                            |   6 +
 programs/uxd/src/utils/math/compute_increase.rs                            |   6 +
 programs/uxd/src/utils/math/compute_shares_amount_for_value.rs             |  20 +++
 programs/uxd/src/utils/math/compute_value_for_shares_amount.rs             |  20 +++
 programs/uxd/src/utils/math/is_within_range_inclusive.rs                   |   9 ++
 programs/uxd/src/utils/math/mod.rs                                         |  13 ++
 programs/uxd/src/utils/mod.rs                                              |   8 +-
 programs/uxd/src/utils/validate_collateral_mint_usdc.rs                    |  41 ++++++
 35 files changed, 2450 insertions(+), 36 deletions(-)
```

---

1. Enable dev vadliations for prod too?

The mint decimal and the different mint checks are disabled in production.

Consider enabling them to make it consistent with the checks for mercurial vault depositories and the identity depository.

In fact, the mint checks for these three types of depositories can be merged.

```
/* repo/programs/uxd/src/instructions/credix_lp/register_credix_lp_depository.rs */
174 | impl<'info> RegisterCredixLpDepository<'info> {
175 |     pub(crate) fn validate(&self) -> Result<()> {
177 |         validate_collateral_mint_usdc(&self.collateral_mint, &self.controller)?;
178 |         Ok(())
179 |     }
180 | }


/* repo/programs/uxd/src/utils/validate_collateral_mint_usdc.rs */
005 | pub fn validate_collateral_mint_usdc(
006 |     collateral_mint: &Account<Mint>,
007 |     controller: &AccountLoader<Controller>,
008 | ) -> Result<()> {
009 |     // Only few stablecoin collateral mint are whitelisted
010 |     // Redeemable and collateral should always be 1:1
011 |     #[cfg(feature = "production")]
012 |     {
013 |         let usdc_mint: Pubkey =
014 |             Pubkey::from_str(b"EPjFWdd5AufqSSqeM2qN1xzybapC8G4wEGGkZwyTDt1v").unwrap();
015 |         require!(
016 |             self.collateral_mint.key().eq(&usdc_mint),
017 |             UxdError::CollateralMintNotAllowed,
018 |         );
019 |     }
020 |     // In development, we can't check the address directly as there are many devnet USDC
021 |     #[cfg(feature = "development")]
022 |     {
023 |         // Collateral mint and redeemable mint should share the same decimals
024 |         // due to the fact that decimal delta is not handled in the mint/redeem instructions
025 |         require!(
026 |             collateral_mint
027 |                 .decimals
028 |                 .eq(&controller.load()?.redeemable_mint_decimals),
029 |             UxdError::CollateralMintNotAllowed,
030 |         );
031 |         // Collateral mint should be different than redeemable mint
032 |         require!(
033 |             collateral_mint
034 |                 .key()
035 |                 .ne(&controller.load()?.redeemable_mint),
036 |             UxdError::CollateralMintEqualToRedeemableMint,
037 |         );
038 |     }
039 |     // Done
040 |     Ok(())
041 | }
```

---

2. Improve redeemable amount range check

Consider adding the following upper bound check. Besides, the `validate` functions for all three types of depositories can be merged.

```
require!(
   self.user_redeemable.amount >= redeemable_amount,
   UxdError::InsufficientRedeemableAmount
);
```

```
/* programs/uxd/src/instructions/credix_lp/redeem_from_credix_lp_depository.rs */
493 | impl<'info> RedeemFromCredixLpDepository<'info> {
494 |     pub(crate) fn validate(&self, redeemable_amount: u64) -> Result<()> {
495 |         validate_is_program_frozen(self.controller.load()?)?;
496 |         require!(redeemable_amount > 0, UxdError::InvalidRedeemableAmount);
497 |         Ok(())
498 |     }
499 | }
```

---

3. Credix PDA validations

User-provided credix accounts are checked against the ones saved in the depository (except `credix_pass`) and will be further validated in the credix contract. Since the depository registration is done by a permissioned instruction, this is relatively safe.

Although their trust root `credix_global_market_state` is included in the depository seeds, consider adding PDA address validations for the following reasons:

- It's unclear how these accounts are validated in the credix contract as its source code is not publicly available.
- There is no instruction to update the saved accounts if some accounts provided in the registration are incorrect, especially if the provided trust root `credix_global_market_state` is not derived from "credix-marketplace".
- The `credix_pass` is not checked in the UXD contract.

```
#[account(,
    owner = credix_client::ID,
    seeds = [global_market_state.key().as_ref(),
             depository.key().as_ref(),
             credix_client::CREDIX_PASS_SEED.as_bytes()],
    bump,
    seeds::program = credix_client::ID
)]
pub credix_pass:  Box<Account<'info, credix_client::CredixPass>>,
```

---

4. Questions regarding amount after precision loss

The following questions are not security related but more about optimizations.

1. deposit_funds

```
/* programs/uxd/src/instructions/credix_lp/mint_with_credix_lp_depository.rs */
169 | let shares_amount: u64 = compute_shares_amount_for_value(
170 |     collateral_amount,
171 |     total_shares_amount_before,
172 |     total_shares_value_before,
173 | )?;
183 | // Compute the amount of collateral that the received shares are worth (after potential precision loss)
184 | let collateral_amount_after_precision_loss: u64 = compute_value_for_shares_amount(
185 |     shares_amount,
186 |     total_shares_amount_before,
187 |     total_shares_value_before,
188 | )?;

236 | // Transfer the collateral to an account owned by the depository
237 | msg!("[mint_with_credix_lp_depository:collateral_transfer]",);
238 | token::transfer(
239 |     ctx.accounts
240 |         .into_transfer_user_collateral_to_depository_collateral_context(),
241 |     collateral_amount,  // collateral_amount_after_precision_loss?
242 | )?;
244 | // Do the deposit by placing collateral owned by the depository into the pool
245 | msg!("[mint_with_credix_lp_depository:deposit_funds]",);
246 | credix_client::cpi::deposit_funds(
247 |     ctx.accounts
248 |         .into_deposit_collateral_to_credix_lp_context()
249 |         .with_signer(depository_pda_signer),
250 |     collateral_amount,  // collateral_amount_after_precision_loss?
251 | )?;
```

Due to the floor divisions,

- `shares_amount` may be smaller than the actual token amount that represents the value of `collateral_amount` collaterals.
- `collateral_amount_after_precision_loss` may be smaller than the actual corresponding collateral value of `shares_amount` LP tokens

We don't know how credix handles roundings. But based on the constraints, when depositing `collateral_amount_after_precision_loss`, can we get `shares_amount` tokens back?

If not, how about `collateral_amount_after_precision_loss + i` if `collateral_amount_after_precision_loss + i <= collateral_amount`? The idea is to get the same `share_amount` tokens with fewer collaterals.

2. withdraw_funds

The requested amount withdrawal amount `collateral_amount_before_precision_loss` and the amount transferred out `collateral_amount_after_precision_loss` may be different.

Does it mean the actual withdrawal amount can be less than the requested amount `collateral_amount_before_precision_loss` and always equal to `collateral_amount_after_precision_loss`?

*** redeem_from_credix_lp_depository**

```
/* programs/uxd/src/instructions/credix_lp/redeem_from_credix_lp_depository.rs */
260 | // Run a withdraw CPI from credix into the depository
261 | msg!("[redeem_from_credix_lp_depository:withdraw_funds]",);
262 | credix_client::cpi::withdraw_funds(
263 |     ctx.accounts
264 |         .into_withdraw_funds_from_credix_lp_context()
265 |         .with_signer(depository_pda_signer),
266 |     collateral_amount_before_precision_loss,  // collateral_amount_after_precision_loss?
267 | )?;
268 |
269 | // Transfer the received collateral from the depository to the end user
270 | msg!("[redeem_from_credix_lp_depository:collateral_transfer]",);
271 | token::transfer(
272 |     ctx.accounts
273 |         .into_transfer_depository_collateral_to_user_collateral_context()
274 |         .with_signer(depository_pda_signer),
275 |     collateral_amount_after_precision_loss,
276 | )?;

358 | // The depository collateral account should always be empty
359 | require!(
360 |     depository_collateral_amount_before == depository_collateral_amount_after,
361 |     UxdError::CollateralDepositHasRemainingDust
362 | );
```

*** collect_profit_of_credix_lp_depository**

```
/* programs/uxd/src/instructions/credix_lp/collect_profit_of_credix_lp_depository.rs */
254 | // Run a withdraw CPI from credix into the depository
255 | msg!("[collect_profit_of_credix_lp_depository:withdraw_funds]",);
256 | credix_client::cpi::withdraw_funds(
257 |     ctx.accounts
258 |         .into_withdraw_funds_from_credix_lp_context()
259 |         .with_signer(depository_pda_signer),
260 |     collateral_amount_before_precision_loss,  // collateral_amount_after_precision_loss?
261 | )?;
262 |
263 | // Transfer the received collateral from the depository to the end user
264 | msg!("[collect_profit_of_credix_lp_depository:collateral_transfer]",);
265 | token::transfer(
266 |     ctx.accounts
267 |         .into_transfer_depository_collateral_to_authority_collateral_context()
268 |         .with_signer(depository_pda_signer),
269 |     collateral_amount_after_precision_loss,
270 | )?;

347 | require!(
348 |     depository_collateral_amount_before == depository_collateral_amount_after,
349 |     UxdError::CollateralDepositHasRemainingDust
350 | );
```

3. E2E tests

We kept getting errors related to uninitialized `credix_pass` in E2E tests. So we didn't have a chance to test the ideas.

```
> Program logged: "Instruction: CollectProfitOfCredixLpDepository"
> Program logged: "AnchorError caused by account: credix_pass. Error Code: AccountNotInitialized.
                   Error Number: 3012. Error Message: The program expected this account to be already initialized."
> Program consumed: 18433 of 200000 compute units
> Program returned error: "custom program error: 0xbc4"
```

In addition, we got this error when running the default test (`tests/test_development.ts`)

```
hi@5d0df655d259:~/project$ anchor test
Warning: cargo-build-bpf is deprecated. Please, use cargo-build-sbf
cargo-build-bpf child: /home/hi/.local/share/solana/install/active_release/bin/cargo-build-sbf --arch bpf
   Compiling uxd v5.1.0 (/home/hi/project/programs/uxd)
    Finished release [optimized] target(s) in 7.58s

Deploying workspace: https://api.devnet.solana.com
Upgrade authority: /home/hi/.config/solana/id.json
Deploying program "uxd"...
Program path: /home/hi/project/target/deploy/uxd.so...
Program Id: 2VsgTXM1WZ2E7vxetsAvT8kPq6Xww831DCV3NxKjG9HH

Deploy success

Found a 'test' script in the Anchor.toml. Running it as a test suite!

Running test suite: "/home/hi/project/Anchor.toml"

bigint: Failed to load bindings, pure JS will be used (try npm run rebuild?)
CONTROLLER AUTHORITY => 🔗 https://solscan.io/account/aca3VWxwBeu8FTZowJ9hfSKGzntjX68EXh1N9xpE1PC?cluster=devnet
BANK => 🔗https://solscan.io/account/Eyh77zP5b7arPtPgpnCT8vsGmq9p5Z9HHnBSeQLnAFQi?cluster=devnet
UXD PROGRAM ID == 2VsgTXM1WZ2E7vxetsAvT8kPq6Xww831DCV3NxKjG9HH


==========================================


🧭 editControllerTest
  ❌ Error: controllerAccount not found
      at Controller.getOnchainAccount (/home/hi/project/node_modules/@uxd-protocol/uxd-client/src/controller.ts:57:13)
      at processTicksAndRejections (node:internal/process/task_queues:95:5)
      at editControllerTest (/home/hi/project/tests/cases/editControllerTest.ts:21:38)
      at Context.<anonymous> (/home/hi/project/tests/test_development.ts:24:5)
  1) Set controller global supply cap to 25mm

  0 passing (279ms)
  1 failing

  1) Set controller global supply cap to 25mm:
     Error: controllerAccount not found
      at Controller.getOnchainAccount (node_modules/@uxd-protocol/uxd-client/src/controller.ts:57:13)
      at processTicksAndRejections (node:internal/process/task_queues:95:5)
      at editControllerTest (tests/cases/editControllerTest.ts:21:38)
      at Context.<anonymous> (tests/test_development.ts:24:5)
```

---

5. "credix_pass" properties

1. `mut` can be removed
2. [resolved] Inconsistent `disable_withdrawal_fee` constraints. This has been addressed in the latest commit.

```
/* programs/uxd/src/instructions/credix_lp/mint_with_credix_lp_depository.rs */
111 | #[account(
112 |     mut,
113 |     constraint = credix_pass.user == depository.key() @UxdError::InvalidCredixPass,
114 | )]
115 | pub credix_pass: Box<Account<'info, credix_client::CredixPass>>,

/* repo/programs/uxd/src/instructions/credix_lp/redeem_from_credix_lp_depository.rs */
122 | #[account(
123 |     mut,
124 |     constraint = credix_pass.user == depository.key() @UxdError::InvalidCredixPass,
125 |     constraint = credix_pass.disable_withdrawal_fee == true @UxdError::InvalidCredixPassNoFees,
126 | )]
127 | pub credix_pass: Box<Account<'info, credix_client::CredixPass>>,

/* programs/uxd/src/instructions/credix_lp/collect_profit_of_credix_lp_depository.rs */
099 | #[account(
100 |     mut,
101 |     constraint = credix_pass.user == depository.key() @UxdError::InvalidCredixPass,
102 |     //constraint = credix_pass.disable_withdrawal_fee == true @UxdError::InvalidCredixPassNoFees,
103 | )]
104 | pub credix_pass: Box<Account<'info, credix_client::CredixPass>>,
```

---

6. Minor minor: wording

```
/* repo/programs/uxd/src/error.rs */
098 | #[msg("The Credix Pass isn't the Depository one.")]
099 | InvalidCredixPass,
100 | #[msg("The Credix Pass isn't the without fees.")]
101 | InvalidCredixPassNoFees,
```