Essential items:

- The URL of a repo containing the source code

  https://github.com/zer0-os/zBanc/tree/dynamic-upgrader

- Please invite the GitHub users **@david-oz** and **@tintinweb** to the repo

  https://github.com/zer0-os/zBanc/commit/8fa0aefea6e915529c30f46ec10db7702c8114c4

- The commit hash to be reviewed (OK if this is changing until March 1st)

  https://github.com/zer0-os/zBanc/commit/48da0ac1eebbe31a74742f1ae4281b156f03a4bc

- A list of the files in scope and out of scope

  contracts/converter/types/dynamic-liquid-token/DynamicLiquidTokenConverter

  contracts/converter/types/dynamic-liquid-token/DynamicLiquidTokenConverterFactory

  contracts/converter/ConverterUpgrader.sol (added handling new converterType 3)

- Documentation describing the intended functionality of the system

  The DynamicLiquidTokenConverter does everything that Bancor's LiquidTokenConverter does, with the added functionality of changing the conversion weight.
  To support upgrading the DLTC, we've followed the same pattern as the LiquidityPoolV2Converter for upgrading stateful converters:
  We've made the DLTC converter type 3, and handled the typed data for it in the ConverterUpgrader.

- A list of the key risks for us to ensure are mitigated

  The DLTC should correctly and securely convert to and from token and reserve even after changing the weight settings.
  The DLTC should successfully upgrade with the DLTC state variables and ConverterBase state correctly moved to the new converter.

  DLTC settings are properly validated and don't lead to dangerous rounding problems when above/below certain values, such as the PPM_RESOLTION

- Names and emails of the key personnel we will work with during the engagement

  Damien Burbine
  damienburbine@gmail.com9

Design and Intended Functionality -

The DynamicLiquidTokenConverter (DLTC) does everything that Bancor's LiquidTokenConverter does, with the added functionality of changing the conversion weight.

The DLTC owner should be able to change the conversion weight by the defined step, and the issuance curve should accurately change in response.

Users should be able to create new DLTCs by accessing the DLTCFactory through the DynamicContractRegistry

Any user should be able to convert between reserves based on the bancor formula and accurately after the weights have been updated by an owner.

The DLTC owner should be able to set its state variables before activation, but after activation only the upgrader should be able to set it (the current commit only allows setting when inactive).

The DLTC owner should be able to initiate an upgrade, and the reserves and settings should successfully transfer.

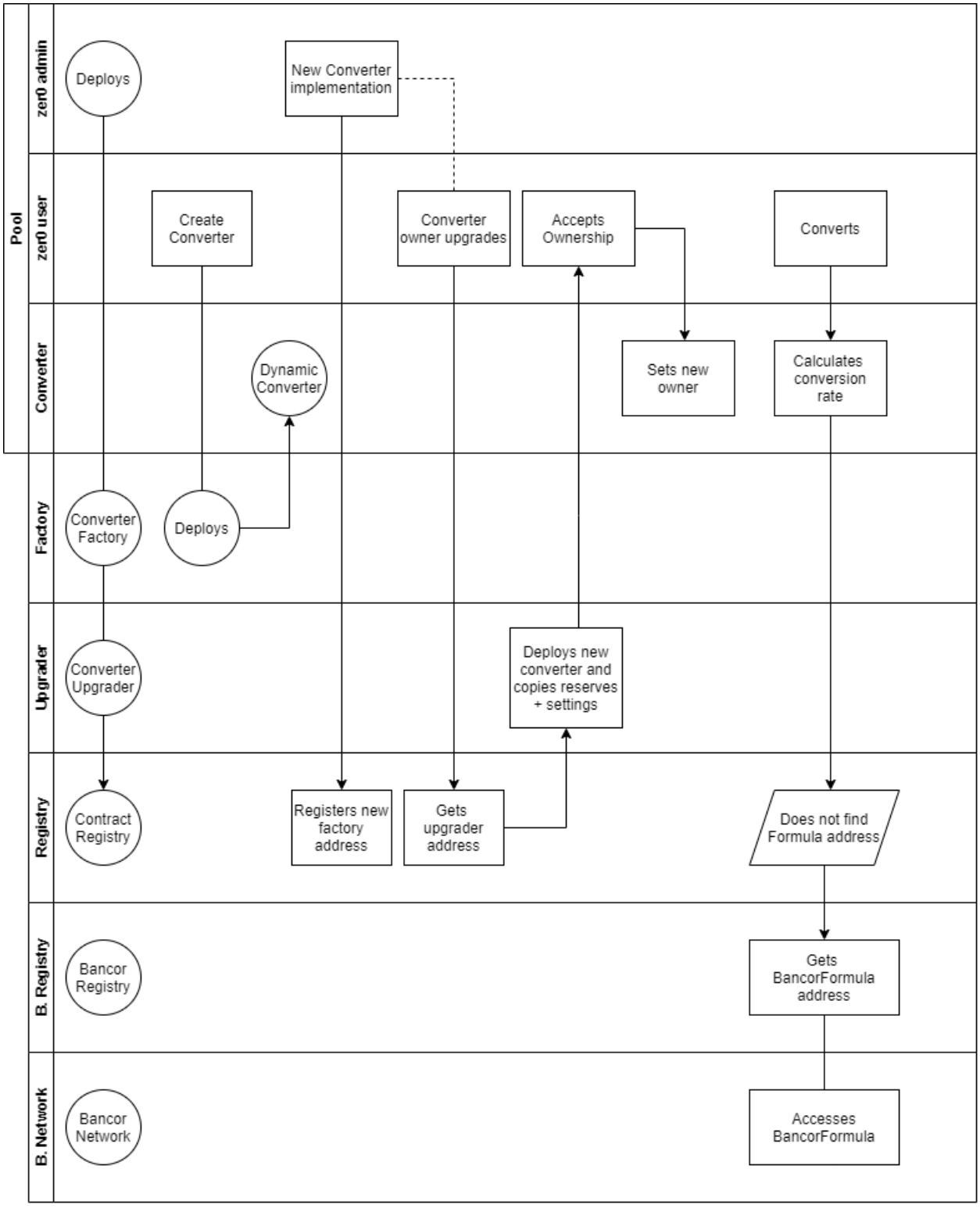
Below is a diagram for the upgrade process for a DLTC. It follows the bancor patterns for upgrading. To handle this, we deploy our own ConverterUpgrader and ContractRegistry owned by zer0 admins who can register new addresses. We do not deploy a BancorNetwork, BancorFormula, or any other contracts in the environment, accessing the bancor deployments instead.

The DynamicContractRegistry (DCR) we deploy has a setting for the bancor ContractRegistry reference, so that a zer0 admin can maintain access to all the contracts there including the BancorFormula - this needs vetted as the DLTCs should not lose functionality due to this maintenance process.

The DCR only holds the ConverterUpgrader and DLTCFactory addresses, any request for an address it doesn't have is instead read from the bancor registry.

DLTCs should not be able to access the bancor ConverterUpgrader or ConverterFactory, which have the same names in both registries.

To upgrade, a zer0 admin sets a new DLTCFactory in the DCR, and then a DLTC owner can call the upgrade() function, which tells the ConverterUpgrader to deploy and transfer reserves and settings.


Pool
zer0 admin
zer0 user
Converter
Factory
Upgrader
Registry
B. Registry
B. Network
Deploys
New Converter implementation
Create Converter
Converter owner upgrades
Accepts Ownership
Converts
Dynamic Converter
Sets new owner
Calculates conversion rate
Converter Factory
Deploys
Converter Upgrader
Deploys new converter and copies reserves + settings
Contract Registry
Registers new factory address
Gets upgrader address
Does not find Formula address
Bancor Registry
Gets BancorFormula address
Bancor Network
Accesses BancorFormula

Updates -
At the commit you started with, the ConverterUpgrader is not quite finished, it does not have the correct factory to deploy the DLTC in the upgrade process, and it does not access our registry. There is a new commit with the finished ConverterUpgrader and DynamicContractRegistry: https://github.com/zer0-os/zBanc/commit/3d6943e82c167c1ae90fb437f9e3ed1a7a7a94c4

I understand you may not be able to fully switch commits, but perhaps you could reference this new commit for at least just the ConverterUpgrader? That would be adequate for us. Hopefully this actually helps more than it delays, as most of this stuff is already in the current commit, except in a broken state that may be difficult to report on.