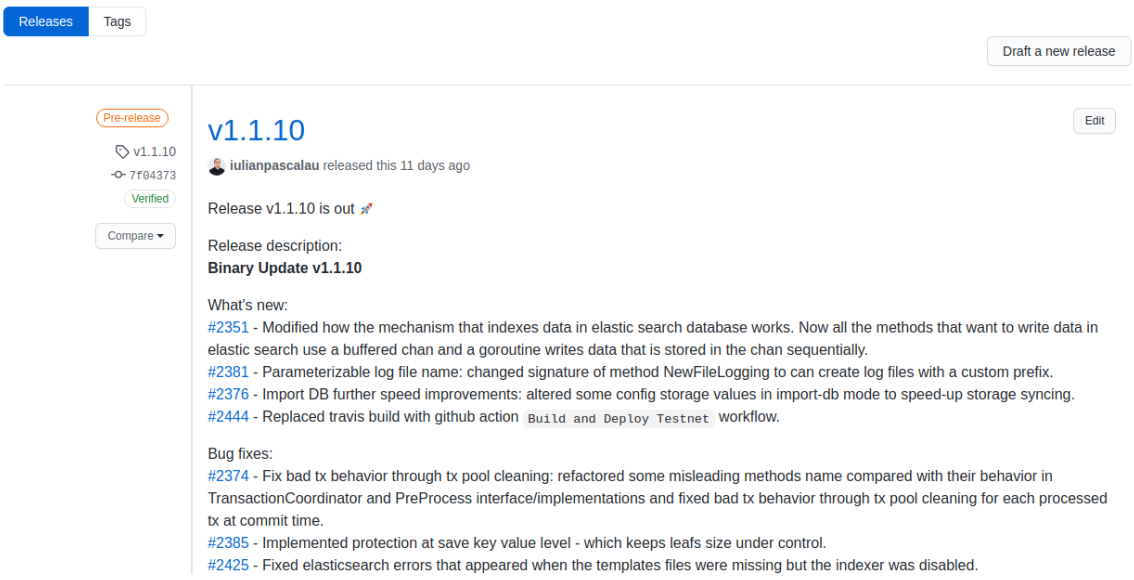
Testnet and mainnet release step-by-step guide

IMPORTANT: do not attempt a testnet or mainnet upgrade while the networks are performing other scheduled maintenance work or tests. Doing so might hinder other work or the stability of the entire network due to unforeseen consequences. Also, a failed update might disrupt, delay or postpone the other critical operations.

Part 1. Testnet release

Section 1.1 mx-chain-go repo:

1. Ensure that the required `rc/*` branch contains desired features.
   The naming convention for the binary is: major version will be 1 (until we have some breaking changes, this is left to be decided). The minor version is directly correlated with the most rc branch released. The right-most version number will be incremented 1, 2, 3...9, 10, 11, 12...98, 99, 100, 101… and so on.
2. **IMPORTANT: Ensure to not reuse tags in case of a bad release (branch incorrectly selected, branch updated and such).**
   To check that the tag is not on a wrong commit, anyone can check with the following command:
   `git fetch -t`



Section 1.2 mx-chain-testnet-config:

3. If the release needs to be started from scratch, follow the steps below, otherwise skip to next step (4):
   3.1. Let's assume that we want to start with the binary version v1.6.x (usually the current version of the mainnet). Create a new branch T1.6.x.y where y is the next free number (0 might have been taken already, take some other number, doesn't matter too much).

   3.2. Bring the configuration files from mx-chain-go repo, from the released tag we want to use. Check all the required changes (this step can not be automated nor can I provide guidelines here as the scenario can vary greatly). Pay attention to the total number of keys as that value should be rewritten in the systemSmartContractsConfig.toml file, options MaxNumberOfNodesForStake and NodesToSelectInAuction. Do not forget about nodesSetup.json, startTime option. Allow at least 12h for the network to update to the new release.

   3.3. Using a tool or writing by hand, generate the nodesSetup.json and genesis.json files.

4. Create a new branch starting from `master` branch, create a new branch called T1.6.z.0, assuming we just released the z version on mx-chain-go repo.

5. Apply changes in the new branch T1.6.x.y / T1.6.z.0. Pay attention especially to epoch change flags as to match the running mainnet network. Change binaryVersion file to match the binary release tag, in our case should be something like `tags/v1.6.x` or `tags/v1.6.z`

6. Announce the testnet validators that a new release will be available but should not attempt to test it until the official announcement is issued. This step is required to fully test the release, including the scripts part.

7. **Important: merge this branch into `master` branch so the next updating release will have the correct keys.**

8. Commit changes in branch T1.6.x.y / T1.6.z.0, create a new release T1.6.x.y / T1.6.z.0 directly released (not pre-release)

9. Test the new release on a fresh machine, the node binary should start (either printing negative rounds when starting fresh or start the syncing process if the release is an upgrade). Open a terminal on that machine, type in the following command as to check the generated number of shards is **3**:
   ```
   curl -s http://127.0.0.1:8080/node/status | jq -
   r .data.metrics.erd_highest_final_nonce jq -
   r .data.metrics.erd_num_shards_without_meta
   ```

Check the version of the compiled binary to match the binary release (T1.6.x or T1.6.z). This can be done using the termui interface.
Failure to do so will result in deploying a wrong binary version.
Related to this check, it is recommended that the mx-chain-go source code will be checked against latest changes from the release tag to ensure that the cloned repo & checkout branch matches with the released version. This check will catch the release bug called “forgot to update binary tag string in binaryVersion file”.

It is highly recommended that a release will be tried upon 4 different nodes, each on each shard. If the sync process stops on a machine or the node does not start, the release should be reconsidered (start the troubleshooting by re-checking the configuration and after that reconsider the written code)

10. Test the new release at least 15 minutes (1-2 epochs preferably) to check and re-check that no incompatibilities found their way.

11. Announce other validators and start the upgrading process.

Part 2. Mainnet release

12. **IMPORTANT: the 1-11 steps are mandatory on each mainnet release. This is to assure us that no irreversible mistakes can happen on the mainnet. Failure to do so is a flagrant breach of the current procedures and should not be accepted by the rest of the team, under any circumstances!**

13. Since we have the binary releases, we can directly start the configuration set-up process.

14. Immediately after announcing the new release on mainnet, merge the relevant branch (usually the release candidate branch) into master branch. Do the same on all satellite projects, if required.

15. After a couple of epochs we can make a new upgrading release (if required).

Section 2.1 mx-chain-mainnet-config:

16. The steps of setting up the config values resembles a lot with the steps from section 1.2 mx-chain-testnet-config with minor changes:
16.1 - step 3 should never be attempted: we will delete the existing blockchain history.
16.2 - branch versions do not start with capital T but with lower v

Release time considerations:

1. We have started planning the next one or 2 major releases in advance with tentative release schedule dates. These will be called ‘planned’ releases and will include new features or activation configuration changes. As for now, these are scheduled in approximative 1 month intervals.
2. Unplanned releases should be avoided because they directly affect 3-rd party validators/integrators.
3. Avoid the activation time for a mainnet release during the weekend or on a Monday. When a release should be carried in August increase the activation period as this month is usually the most unproductive month due to holidays.

Part 3. Troubleshooting

Section 3.1 managing releases & tags:

1. If the release needs to be changed (because there was a release error or a wrong config value happened to pass unnoticed), both the release **and** the tag will need to be deleted. Failure to do so will result in creating tags and releases that are not “seen” as the-most-recent-ones.
2. Deleting a release will be done by clicking directly on the release title:

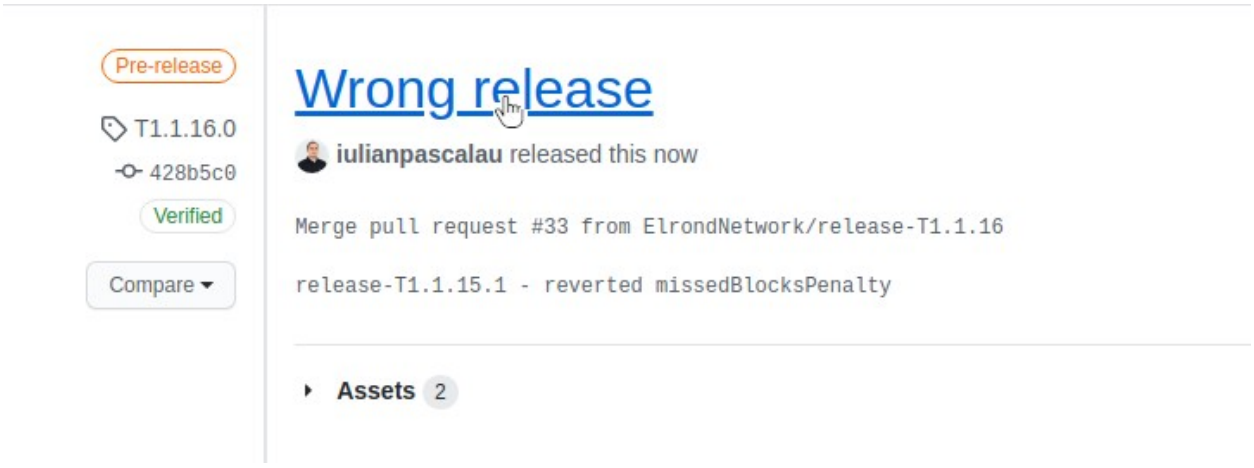


Fig. 3.1.2 - Clicking on the release title “Wrong release”

After clicking the release title, the button called “Delete” will be pressed.

3. Deleting the release tag will be done by clicking directly on the tag title:

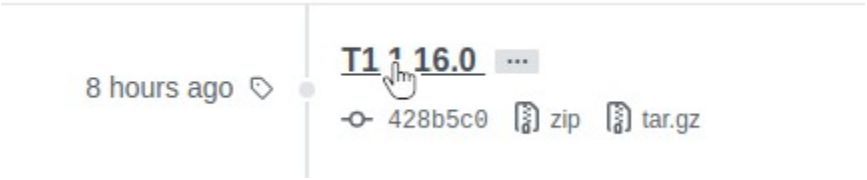


Fig. 3.1.3 - Clicking on the tag title “T1.1.16.0”

After clicking the tag title, the button called “Delete” will be pressed.