Updating Safex from Monero

Goals

- Provide the reader with the precise process of how to replicably merge new changes from Monero to the Safex codebase

On This Page:

Initialization

First, clone safexcore to your own github account, and clone it.

```
git clone --recursive https://github.com/{USERNAME}/safexcore.git
```

Next, switch to the repository folder and initialize the

```
cd safexcore
git remote add origin https://github.com/{USERNAME}/safexcore.git
git remote add upstream https://github.com/safex/safexcore.git
git remote add monero-upstream
https://github.com/monero-project/monero.git
git fetch origin
git fetch upstream
git fetch monero-upstream
git checkout -b develop origin/develop
git checkout -b safex-develop upstream/develop
git checkout -b monero-master monero-upstream/develop
```

Then on the **Monero master branch**, get the latest git tag.

```
git checkout monero-master
git pull
git tag --sort version:refname
```

Let's assume the last tag printed on the screen (also the latest one) is **v0.12.3.0** - we then need to incorporate all changes from that release that we don't have in our own. First, we fetch a list of those changes:

```
git log --pretty=oneline --since="20-04-2018"
v0.12.3.0...upstream/develop > diff.txt
less diff.txt
```

As a sidenote, Monero also uses specific branches, for clearing up bugs before releases, but they all seem to be synced with the master branch.

The `diff.txt` file that is the result of the previous line should be the same as `diff.txt` this one:

```
git fetch monero-upstream
git checkout -b monero-release-v0.12 monero-upstream/release-v0.12
git log --pretty=oneline --since="20-04-2018" HEAD...upstream/develop >
diff.txt
less diff.txt
```

If uncertain that you're missing a pre-release bugfix, you can always compare the two branches' `diffs.txt`'s quickly by comparing their hashes.

```
md5 diff.txt
```

Now that we definitely have a difference between the Monero's master branch and our own contributions, we need to shortlist that list of commits to something we are likely to want to integrate. That means filtering the commit messages by certain keywords. And it's done like this:

```
git log --pretty=oneline --since="20-04-2018"
v0.12.3.0...upstream/develop | grep -i -e blockchain -e hash -e crypto
-e merge -e protocol -e malloc -e alloc -e alloca > selected_commits.txt
less selected_commits.txt
```

Be sure to **copy the file `selected_commits.txt` to a safe location**. It shouldn't be part of the `monero-master` or a release branch, and you'll need it later. It would be useful to compare it with the `diffs.txt` at this point, just as a sanity check. Otherwise, you won't need `diffs.txt` going forward.

```
mv selected_commits.txt ~
rm diffs.txt
```

This provides us with a list of all relevant commits to work with, plus perhaps some commits from our development branch, **which we will ignore**, of course. We have to go one by one, **from the oldest one forward**, i.e. from the end of the file toward the first line, and decide which changes to merge.

Start after the fork commit `8fdf645397654956b74d6ddcd79f94bfa7bf2c5f`, or after the latest commit that has been added or rejected to the safex develop branch.

Integration

```
git checkout monero-master
tig log 03ff3be10ef092aa6129722e99ffd75a7779c1f1 -1
```

This will give you the information about the specific commit. This is especially useful if the commit is a merge, where getting information this way is **far preferable** than getting it from GitHub immediately. If it is indeed a merge, then go through each commit on GitHub, which provides a nice overview. GitHub is best for commit-level overview.

When you're ready to integrate proposal changes, you need to make a branch for that batch of changes on your own fork of `safexcore`, which is `{USERNAME}/safexcore`.

```
git checkout -b xmr_batch_1
git reset --hard upstream/develop
mv ~/selected_commits.txt ~/safexcore
git add selected_commits.txt
git commit -m "Added selected_commits.txt"
git push origin xmr_batch_1
```

Let's assume that we've decided to see whether the commit `b1a9e97b2dab7a843571c674b242926d51362a6e` should be merged onto our development branch.

1. If you don't want to add a merge/squash/commit from Monero to Safex, then add `[REJECTED]` before that commit's message in `selected_commits.txt`
2. Make sure you've properly reviewed it on GitHub https://github.com/monero-project/monero/commit/b1a9e97b2dab7a843571c674b242926d51362a6e
3. Cherry-pick the changes made in that commit onto your development branch.

   ```
   git checkout xmr_batch_1
   git pull
   git cherry-pick b1a9e97b2dab7a843571c674b242926d51362a6e
   ```

   NOTE: NEVER CHERRY-PICK WHOLE MERGES OR SQUASHES. ONLY INDIVIDUAL COMMITS.
4. Resolve the merge conflicts
5. Build & Run Tests
6. Solve all bugs and problems
7. Proceed to next commit ONLY when all changes work smoothly
   NOTE: When you're done with a batch of changes, **DO NOT SQUASH THEM**. Much information will be lost, and the above process will have to be manually repeated in case of any errors that are hard to track.
8. Add `[IN BRANCH]` before that commit's message in `selected_commits.txt`
9. Repeat from 1.

Once enough commits are added to make a reasonable batch, then comes the next phase for that group of commits:

1. Create a pull request onto safex/safexcore develop branch.
2. Change `[IN BRANCH]` to `[IN PULL REQUEST]`
3. When the commit is integrated, change `[IN PULL REQUEST]` to `[INTEGRATED]`

Keeping in Track with Develop

If there have been any recent additions to the `develop` branch in `safex/safexcore`, then you should update your branch ASAP, using the following commands.

```
git checkout safex-develop
git fetch
git pull
git checkout xmr_batch_1
git rebase safex-develop
```