Thunderbird

Build & Release

How It’s Done

05.31.24 Wayne Mery — Release Manager, Daniel Darnell — Build Engineer

Table of contents

Helpful Terminology

1. **Release Channel**
   Used to determine which release train you receive updates from (Daily, Beta, Release/Monthly, ESR)
2. **Balrog**
   Web app used to control release sign-offs and update rates for each release channel
3. **CDN**
   Content Distribution Network – Network of servers that efficiently distribute Thunderbird updates to users
4. **Taskcluster**
   Task execution framework used to build and release Thunderbird
5. **Treeherder**
   Web interface for Taskcluster job steps
6. **Ship-It**
   Web app used to create releases from a given build using Taskcluster
7. **Smoke test**
   Manual tests to verify that a build is ready for delivery to users
8. **Uplift**
   Another term for backport, which is the process of applying a specific software patch from a newer version or channel to an older version or channel

The Pre-Release Process

Uplifting New Bugs

How new patches get from Daily to Beta, ESR, and Release

Uplift Request

Uplift is usually requested by the patch author

- Request is assessed and approved/denied by release manager
  - If a patch needs to be modified for a different release channel, a modified patch is requested from the author

Uplifting Approved Patches

On or before the release date of a beta or ESR, patches approved for a release channel are grafted into the appropriate branch

- If a patch cannot be merged cleanly or with a simple 3-way merge correction, a patch revision is requested from the author

After Uplifts

The newly-updated branch of Thunderbird is versioned for a new release

- The branch is pinned to the newest corresponding version of Firefox
- Either automatically or manually, the version number is updated
- Links to each uplift on the new branch are posted on bugs included in the release

Writing Release Notes

How Release Notes are Picked, Written, and Organized

When Notes are Written

Several factors determine whether release notes are written

- Does the bug impact end users?
- Is the bug a strictly internal change?
- Is the bug routine upstream synchronization?

How Notes are Written

Notes are ideally written to be informative but accessible to all users

- Describe the problem succinctly and in passive voice
- Organize the notes by various traits (from most to least important)
  - Type - new, changed, or fixed
  - Area - mail (and news), address book, chat, calendar, or user interface
  - Feature - IMAP, CalDAV, global search, etc.

Release Cadence

How often each release channel sees a new release

Release channels each get updated at different intervals:

- Beta – Roughly twice a week
  - Merged once per month from daily
- Release – Once per month, with subsequent point releases for approved bug fixes
  - Merged once per month from beta
- ESR – Once per year, with subsequent releases for approved bug fixes
  - Merged once a year from beta

The Release Process

Life Cycle of a Release Candidate

How a build becomes a release candidate, and becomes a release

Promoting

Takes a build and creates and signs release candidate for smoke testing

- Builds L10n (and langpacks)
- Build is pushed to candidates directory of FTP and to “*-localtest” update channel
  - This is how smoke testers acquire the build
- Email is sent to appropriate lists with release notes

Pushing

Pushes the newly-promoted build to the CDN, all but releasing the candidate

- Build is now available on FTP as a new release
- Release notes are made public

Shipping

Ships a release to the public

- Files are copied from the "candidates" directory to the "release" directory
- The release channel rule in Balrog is updated for the new release
  - Builds must be signed off by release engineer and release manager
  - This allows regular users to get a small download the release as an update

Testing a New Release

How new candidates are tested

Smoke Testing Email

Smoke testers are notified that a new candidate is available to test

- Testers acquire build from “<channel>-localtest” release channel or FTP
- Once tested, results are sent back to release manager
- Betas are usually shipped and released same-day, while ESR receives more testing prior to shipping

Checking Documentation

If a candidate passes smoke testing, release notes and security advisories are checked for public availability and accuracy

Merge Day

What happens when we need to sync branches (daily to beta, beta to ESR, etc.)

Close Trees

The trees of the branches being merged from and merged to are closed to prevent changes

- Email is sent out to keep relevant mailing lists apprised of merge progress

Perform Merge

The “from” branch is merged into the “to” branch using build automation

- A dry run is performed first to detect unforeseen issues

Open Trees and Release

If all goes well, the trees are reopened and the new release proceeds as a normal release

- Major version number of branch is increased
- Message is sent notifying lists that the merge has been completed

The Post-Release Process

Monitoring Release Feedback

How feedback is gathered and potential issues are recognized

Various feedback and metrics are followed after a release to see what is being reported and determine if something needs to be done about potential issues:

- Crash rate of new release
- Newly-filed bug reports
- Support requests at https://support.mozilla.org/
- Distro (Linux) feedback
- Social venues (reddit, mastodon, etc)
- Channel support group at https://thunderbird.topicbox.com/groups/beta
- https://stats.thunderbird.net/
- https://support.mozilla.org/kb/thunderbird-beta

Common Complications

What can go wrong and how we remediate it, and other fun things

Zero Days

... is when Thunderbird or Firefox team identifies an active threat - “in the wild”, highly rated security issue - which needs immediate fixing. In such cases we attempt to develop a patch, test, build and ship within 24 hours.

Merge Conflicts

When a patch is written on top of a particular branch of Thunderbird, it may not merge cleanly with other branches. In simple cases, the merge conflict is resolved by hand. For more complex merge issues, the patch must be revised by the author.

Build Failures

If a build fails because of a simple server issue when running a task (HTTP 500), the task is simply rerun. Otherwise, the issue is assessed and a patch may need to be modified then re-uplifted. This requires a new build.

Canceling Builds

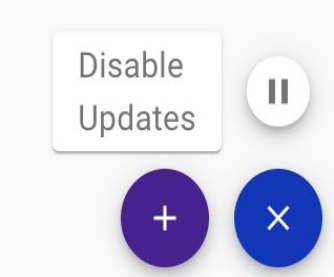
If a build candidate turns out to be non-viable for release, the release process must be canceled. In this case, the current phase of release is halted (if needed), and the release is deleted from Ship-It.

Common Complications (cont.)

More of what can go wrong and how we remediate it, and other fun things

Emergency stop of updates

Balrog “Disable Updates may be used quickly stop all updates (both manual and automatic), no sign off required.



Adjust update rate for special conditions

Balrog is used to adjust update rate for desired objectives, or to even completely disable automatic updates.

We also prevent updates for OS versions which are EOL.

Summary Timeline

1. (relman) Prep: ensure security advisories are prepared, assess channel's current quality and future needs, seek needed patches, evaluate and approve appropriate patch uplift requests, (releng) build/CI fixes and improvements
2. (releng) Apply approved patch uplifts, or do a merge (close trees, perform merge, open trees),
3. (releng) Pin to newest corresponding version of Firefox, set Thunderbird version number, update bugs with patch uplift information, write draft release notes
4. (relman) Use Ship-it to create new version for Taskcluster and promote the build, (releng) fixes anomalies as needed
5. (relman) Inform smoke testers of available candidate build, (testers) exercise build, (relman) assess feedback
6. (relman) Push build to CDN, (releng) fix any anomalies
7. (relman) Ship release, set Balrog update rate, (security) make advisories live, (releng) make release notes live
8. (relman) Monitor release feedback/health, seek corrections in case of poor health, adjust update rate

releng = Release Engineering relman = Release Manager

Links and Further Reading

- Download Channels: ESR, Beta, Daily, default release
- Thunderbird's release and events calendar (closely follows the Firefox release schedule)
- Thunderbird Release Notes
- Thunderbird CI Docs
- Treeherder
- Thunderbird statistics
- Thunderbird crash statistics

Thank you!