The official policy, approved by Tiki Admins, is here: Version Lifecycle. The ideas below were initiated by a community member (mikeua) and after debate, were withdrawn.

PROPOSAL WITHDRAWN

- Let's delete this page.
  - We'll consider the proposal withdrawn but we will not delete the page because the brainstorming has value. We improve ideas and plans by taking in account wisdom and experience of the past. You proposed things and some answers / reasoning has been offered. The situation can change, and the arguments/proposals can evolve.

One Version of Tiki

- Explore the possibility of switching to only one main version of Tiki, an STS on a 3-month update cycle (security patches any time), plus one leading Beta on a 'Rolling Release' cycle (so eliminate the LTS's and all others) as one part in an overall plan to increase quality and sustainability of Tiki as well as grow the Community of Developers and non-Developers.
- This was informally discussed on the Dev List in December 2020
- This was a bit more formally addressed in the January 2021 TRM, 'Second Hour Topics' point 4,5 and 6

Why?

- **Make better use of current developers time**
  - There does not seem to be enough developers in relation to the current philosophy of Tiki ('Tiki does it all') (statistics should be added)
    - I don't think there's a direct relationship between the philosophy of "Tiki does it all" and the number of developers. Many/most of Tiki's features are mature and don't need a lot of maintenance.
  - There seems to be too many past versions of Tiki that need supporting, even if just for small security fixes, in relation to the number of developers available.
    - I think having no LTS version would cause problems for organizations that want the stability of long term support.
    - Now, people can have a supported Tiki with 1 major upgrade (not easy) every 4 years, and 8-10 minor (easy) upgrades during this period. Instead, you are proposing 4 major upgrades per year, so this would mean 16 major upgrades + minor releases (perhaps 5-10 over 4 years, maybe more). So more than double the number of upgrades. Because it would be more incremental, some of these major releases would be easy upgrades but without an LTS, the community would never know if a major disruptive upgrade is lurking a few months away. With an LTS, we can concentrate all our most disruptive changes (like Bootstrap 3 and Bootstrap 4) in a post-LTS. And users have a choice between stability and innovation, for each Tiki instance. Also, 3 months is not enough time to make major changes like moving to Bootstrap 5.
      - In my not so humble opinion of someone who has been doing this since 2003: It is flawed logic to think that by removing LTS versions, it would magically concentrate all the energies and make things more stable. The reality is that most of the community would not follow and we'd end with a multitude of Tikis in various unsupported versions. Now, at least, we are converging the LTS users together and those versions do get support. Maintaining LTS versions is overall for the community at least an order of magnitude less work than what you propose. So doing this (removing LTS versions) would massively hurt the community. I can tell you right away that this not going to happen. We can tweak the number of months for STS (was 6 for a while and now it's 8). It could be faster or it could be slower but the concept of an LTS is here to stay. The LTS is now every 3 versions. It could be more or less. The total 5 year support could be tweaked as well. We

just need to carefully weigh the pros and cons.

- **Improve Quality**
  - There have been a number of reports on the Dev list and Forums that seem to indicate a preference by the current Tiki Admins for more Features over higher quality of existing features
    - I don't think this is actually the issue. As discussed before, Tiki often gets a new feature because the client of a Tiki consultant needs the feature and has a budget for to pay for it, and so it gets added to the Tiki code. It's not like there's a preference or a choice being made by Tiki admins or developers. Of course bugs in existing features need to fixed — this is something that bothers me and probably all of us — but the issue is who has the skills and the time to do it as a volunteer. It's even possible that if the coders involved in Tiki decided, ok, let's not work on client-requested-and-paid-for new features, and just fix existing bugs, how do those developers make a living? They'd have to get a job/clients doing something other than Tiki and quite likely have no more time or interest in the project anymore. On the other hand, developers that become part of the community because there's a client paying for a new feature also tend to fix other bugs in the code apart from that feature.
    - Mike Finko: This is again flawed logic that by preventing people to do what **they** want, they are magically going to do what **you** want. People add more features because it increases Tiki's value to them and thus, their commitment to the platform. And even if it did work: who will decide if the features are good enough now, so we can start adding more features? Reasonable people can have a very different opinion of if a feature is good enough. That would be a political nightmare and cause huge tensions in the community. So people would just create their features on their own forks, and we'd end up being a super fragmented community. So again, not going to happen because it would cause massive damage to the community.
  - currently, there is no 'Quality Assurance' team or other methodical and consistent quality program outside of limited testing (e.g. Dogfooding)
    - Mike: You are most welcome to participate to this or any team.
    - Quality Team has been stopped
    - replaced by the Continuous Integration Team and Developers Team
      - Well, this is not a direct replacement. The Quality Team would review backports to reduce the odds of regressions. I don't think this is a priority now because regressions in the LTS branches are rare (Backporters tend to be careful; Our community is more mature and the guidelines are followed; We now have Git for easier source management; We dogfood on *.tiki.org sites with daily updates; We have some automated testing to detect syntax errors)
- **Provide Focus and Direction**
  - the current model of Tiki, 'Funding Features' sounds great in theory, but ends up pulling Tiki in many directions simultaneously, resulting in a 'graveyard' of unfinished or buggy 'Features'
    - I don't really see the situation this way. What conflicting directions is Tiki being pulled in, and where is this graveyard of unfinished/buggy features? I would say that the features that are funded by a Tiki consultant's client are likely to be the least buggy because they are recent work and they must work to meet a business requirement. If there's a problem with the feature working in another Tiki instance, that might be due to configuration or context differences.
    - Mike: It is true that some features funded by clients end up not being used by them, so this is a risk of the feature becoming an "orphan". For example, the client who sponsored SAML never used it. . But
      1. It is a very small proportion. Clients who sponsor a feature usually use and fund work to keep it in working order. In this case, the client was no longer there to maintain the SAML integration.
      2. These features (like SAML) are on the general roadmap of Tiki, so if they are

popular/important enough, someone else will step up.
3. All the sponsored features I can think of are either optional or useful to all. So either you benefit, or you don't activate and there is no negative impact on you.
4. They obviously plan to use the feature, so we don't know in advance which ones will end up "orphaned"
5. If you make a list of the "graveyard" features, I will tell you which ones were sponsored (even if it's not through EvoluData, I tend to know about who sponsored what), and we will let the data tell us.

- Mike: So who would decide what is accepted and what is not? Again a potential community management quagmire. The features will still get done for clients but they won't be contributed to Tiki. So we'd fragment the community.

- Goal: create not only a general, long term Roadmap for development of Tiki but also short and medium term goals, and, as mentioned above, for developing a Community of non-developers.
  - note: point #5 by Jono Bacon's article "Redemption: My Community Management Career Mistakes (and How To Avoid Them)":
    - ***#5. The Feature Suggestion Snafu – many communities have been tempted by the idea of letting your community members suggest product features. Be careful though: it can result in a mishmash of unmet expectations and problems the size of a small city that are difficult to ever fix. Ooof, this one sucked. link***
      - This doesn't seem to apply to Tiki because, generally speaking, while it's great for Tiki community members to suggest product features, these aren't likely to be implemented unless those community members are coders or can convince a coder to volunteer time, or can pay a coder to implement it. And there are precedents in Tiki to remove features that aren't maintained because the creator left the project.
      - Mike: Some community members can set all the goals they want and make all the plans they want. Heck, I do it all the time. Do these because reality? Only if someone actually does the work. This is a do-ocracy. So the question you need to answer: what incentives and business models can lead to more collaborators, sustainably?

Rolling Release Definition

- one Tiki version that is updated for new Features on a regular schedule - 3 months? 6 months?
  - or is this really just an STS?
- security and bug fixes can be applied at any time

According to Wikipedia https://en.wikipedia.org/wiki/Rolling_release:

❝

> *Rolling release, rolling update, or continuous delivery, in software development, is the concept of frequently delivering updates to applications. This is in contrast to a standard or point release development model which uses software versions that must be reinstalled over the previous version. An example of this difference would be the multiple versions of Ubuntu Linux versus the single, constantly updated version of Arch Linux.*

This matches the common industry usage of the word in my experience and is not what is suggested in this page. Hence the big misunderstanding over what we were talking about during the January TRM.

This page seems to suggest a shortened cycle for STS Tiki releases from the current 8-month cycle to a 6 or 3 months cycle. Is this correct?
It seems that in order to meet *Rolling Release* criteria we would drop the public Tiki versions, a bit like running the "master" branch or a close copy of it, taking all predictability away from end-users. That's obviously what firefox does, but they have way more quality assurance resources than we do.

- thanks for adding the 'Rolling Release' definition, J-M, I should have started with this (I'll change the title of the page) - in any case, the idea was to have only one Tiki version + a leading Beta. So the one main version could be an STS on a 3 month update cycle and the leading Beta could be 'Rolling'. No other older versions, no LTS's - there just does not seem to be enough developers (but, need statistics)

We already have a rolling release: Daily Build

Points of why this cannot be done

- At the January 2021 TRM, I believe it was mentioned that there are factors outside of Tiki's control that need to be addressed, like php version
    - I would argue that if I can change my php version as a non-server specialist, I would guess anyone could. I believe the only issue here is that a message, warning or guidance should be displayed so that all external changes can also be made.
        - This is not always available. Or there can be a lag. Examples:
            - As of 2021-05-13, PHP 7.4 and 8.0 are not available from https://www.softwarecollections.org/en/scls/?search=php
            - show.tiki.org The original show server can only support one version of PHP. Since we need different versions of PHP for different versions of Tiki, it was required to start another server (Show2)
        - Mike Finko: Can you pick your database version? I'll bet that no. And thus, if we increase the minimum requirement for the database, you'll be unable to upgrade. So to avoid pain to the community, we'd always need to stay with old versions. With current model, we can and do aggressively increase the requirements and take advantage of the new features. And community members can stay in LTS if they can't upgrade.
- at the January 2021 TRM, I asked how does 'Slack' or 'Salesforce' manage to only have one version, and I believe I was told that this is not comparable as they are SaaS on a continuous release cycle
    - Indeed not directly comparable as they are not deployable to any server. It's a hosted service and they control all the software.
    - the world is used to these types of models, and, thanks to them, end users are used to frequent (and irregular) change of their interface, making an LTS obsolete
        - LTS is not obsolete, as it's critical to some use cases. Red Hat's main business model is around LTS. The majority of the Tikis I host are on LTS because it saves time and offers more stability to projects. If you don't like it, don't use it, but LTS is here to stay.
    - I do not see how this is different from a 6 month Rolling Release (e.g. new version every 6 months), if anything, a structured release cycle like 6 months is something users can accept and actually better since their GUI will not change randomly - which leads me to believe a shorter 3 month Rolling Release cycle would be better.
        - Mike: if you want to run your Tikis from the Daily Build, you can. I predict you will give up within 1 year because you will gain experience that it's not productive.