Freedom Respecting Technology Definition
bringing about the next generation of Open Source, Free Software, Open Knowledge, Open Culture, and Technological Freedom

definition version: 1.0.1
release date: 2023-06-30

note: This is (generated from) a self contained (though very sloppily/unidiomatically) handwritten HTML file without any other embedded resources and can thus easily be saved for offline reading. It's rather annoying that mobile browsers don't have HTML saving functionality but this is a defect with them not WHATWG standard HTML as an open file format. We'll likely take steps to address this later. Copy and share as widely as possible but don't make modifications without consulting the authors first.

author information

initiator, primary definition author, and benevolent leader for life:
Georgiy Treyvus - makesourcenotcode the_swirly_thing gmail le_pointer com

reviewers, coauthors, supporters, and other helpers:
{insert (nick)names, pseudonyms, and/or email addresses one per line as desired by contributors here}

preamble

Technology in various forms is influencing if not outright controlling ever more aspects of our lives with no signs of slowing down. Both on an individual and collective/collaborative/community level we must understand and control the technology we use or it will control us. The latter may happen by itself in the event of strong artificial intelligence. More likely it may happen by the hands of the few developers who truly understand it and who have a depressingly high probability of being employed/bribed/coerced by unethical corporations and/or regimes. However the latter scenario comes about it's absolutely unacceptable.

Thankfully we are not alone in our sentiments here. Long before most of this document's authors were born many people started work in this direction. A prominent example is Richard Stallman who formalized these sentiments as well as certain good cultural tendencies he had seen with regards to the sharing of knowledge which were slowly disappearing into what became the GNU Project and the Free Software movement more generally. This also inspired at least indirectly other movements of free (as in freedom) culture/technology including but not limited to Open Source and Creative Commons. All in all these movements have done a tremendous amount of good and are huge steps in the right direction. Ultimately it is on the shoulders of these giants upon which we stand as we write this document in an attempt to progress yet further.

Herein we attempt to fully articulate and address longstanding and growing frustrations we've been feeling for years if not decades. These originate mainly with regards to Open Source Software, to a lesser extent Free Software, and extend well beyond just software into technology more generally.

While many FOSS(Free and Open Source Software) projects are often made in good faith and we often can't express in words how deeply we appreciate some of them the fact is that the vast majority of (sometimes very promising) FOSS projects fail badly to properly deliver on certain critical aspects of openness, accessibility, and technological freedom. In spite of being fully compliant with things like the GPL(GNU General Public License), FSD(Free Software Definition), and/or OSD(Open Source Definition) thus being theoretically free many FOSS projects fail in making technological freedom practical. Unfortunately there are a lot of mostly artificial barriers that (usually unintentionally) stand in the way of many interested people (including very skilled competent ones) exercising the crucial freedoms of using, studying, understanding, modifying, copying, (re)distributing, further developing, and contributing back to FOSS projects in a truly free, autonomous, and self sufficient manner.

Thus we seek to develop this Freedom Respecting Technology Definition (or FRTD for short) driven by the principles of radical understandability,

radical discoverability, radical accessibility, radical open knowledge, and radical self sufficiency. (A crucial consequence of that last bit is that it can, should, and hopefully ultimately will also lead to healthier and significantly more resilient individuals and local communities.)

We're aiming to carefully study, formalize, and explicitly forbid as many of the devastating failure modes we've seen in FOSS projects at both the user and developer level as we can. (We're also aiming to blur if not outright eliminate depending on context the mostly if not entirely artificial boundaries between users and developers.) We also aim to carefully study, highlight, formalize, and explicitly require at least the bare minimum set of best practices which some FOSS projects have engaged in (even if only accidentally in some cases) that have given end users real practical technological freedom. Though we won't be able to do it perfectly we aim to have something that as much as possible formalizes and captures the very essence of technological freedom. We're aiming for something which can guide ourselves and other well meaning developers of technologies including but not limited to software, firmware, hardware, protocols, technical specifications, algorithms, scientific research, knowledge, media (informative and artistic), and combinations thereof(which is what most things are) towards building a truly free and healthy world.

While this document is intended primarily to address problems with Free Software and Open Source Software a lot of the issues discussed here pertain to all software and to technology more generally. Thus we hope this will be read by, of interest to, and acted upon by a wide and diverse audience. We feel that this document can help even developers of closed source software make many aspects of it better as well as provide good food for thought to more moderate movements such as Fair Source. This is because as one often hears freedom is not free whether the cost comes in the form of effort, labor, time, money, or something else. For a technology to support freedom even partially let alone completely it must have certain qualities. Not trying hard to take away people's technological freedoms is not enough. Active steps must be taken to add qualities that create and preserve freedom. Some side effects of technologies having some of these qualities are better user and developer experiences as well as better internal design.

In addition to this definition we intend to release a few FRTD compliant technologies to the public both for their personal utility and more crucially for peer review. These projects are intended to demonstrate and capture the spirit of Freedom Respecting Technologies (or FRTs for shorthand) in ways that even the best possible version of this document will likely fail to. We want people to feel in a hands on manner the difference between Freedom Respecting Technologies and typical FOSS. We want people to feel it viscerally. These example projects will almost certainly come in the form of software as things like hardware for example are largely outside of our current areas of expertise. Help realizing FRTs in other mediums is enthusiastically welcomed and would be deeply appreciated.

We want to test the approachability, discoverability, learnability, accessibility, and understandability of these sample projects on a wide variety of people from both the user and developer perspectives. We want to assure that skilled developers aren't adversely affected by any of the requirements of the FRTD. We want to see if novice developers find the projects approachable for hacking on. We want to see how quickly and easily a random user can cross into developer territory to modify a given FRT to be more in line with their desires. Again being able to do this in theory is not enough. It must be practical.

Finally we hope that this definition and the technologies we release will be the beginning of a movement at least as large and successful as FOSS was. We dream of a world where people can combine FRTs realized in various mediums like hardware, software, etc into full solutions to their technological needs and wants.

main thesis

Failure of FOSS projects to provide the crucial technological freedoms we seek stems from their failure to be reasonably understandable to people (even sometimes ones with significant expertise) in a reasonably self sufficient manner. Understandability and self sufficiency are the two absolutely necessary prerequisites for technological freedom. Hence we examine what it takes to provide both.

Open Source is an absolutely necessary but often insufficient subset of Open Knowledge which is the real goal and the key to understandability. The crucial but still ultimately narrow focus on source access often distracts us from focusing holistically on proper access to the full Open Knowledge Set which is what truly matters.

For example in the case of software another important element of the Knowledge Set is information exposed by the UI(user interface) in the environments where that software can be run. Another even more important element is both the built and source forms of the Support Information like any official written documentation, diagrams, instructional/demonstrational videos, and so forth that may exist.

Indeed a lot of what this document will focus on is this Support Information. Most systems are useless without it from both a user and developer perspective. Depending on how self describing the components of the technology/system in question are the amount of needed Support Information will vary. However for all but the simplest technologies it will be necessary. Happily many (though not a majority of) FOSS projects do a somewhat decent job of providing this information in circumstances where most reasonable people would deem it necessary and/or desirable. The main issue we have is with how it's not accessible in a reasonably self sufficient manner.

Ultimately understandability comes from the full Open Knowledge Set being as ubiquitous and easy to access and cleanly copy as humanely possible in both built and source forms. At the very least this should be possible in wholesale fashion and also ideally in more targeted subsets which aids certain kinds of accessibility we'll touch on later.

While we're on the subject of open knowledge it is important to be cognizant that knowledge is not quite the same thing as raw factual information. Knowledge is contextualized information and the FRTD is about Open Knowledge and not Open Facts. This context is best thought of as some meta facts that highlight/enumerate the existence of all the other facts either directly or recursively and help them make sense in some kind

of conceptual framework. Often a very small bit of carefully placed context can take a system from absolute zero to literal superhero.

The other key to freedom is technologies enabling self sufficiency. It's crucial to maximize this as much as possible by minimizing the things one is dependendent on to study, use, and further develop a technology.

One very important thing this means is the full Open Knowledge Set being cleanly enumerated and easily available for offline use. Not just the main source code and executables as is often the case. Withholding any other parts of the full Open Knowledge Set from easy offline access/study/use is absolutely no different than withholding the main program sources. Whether they are given away for free or sold (for mere reproduction cost or some not extortionate profits) we don't care. The full Open Knowledge Set must be usable in a self sufficient manner in any FRT.

Critically special emphasis here is on built versions of the Support Information Set. At the very least it must be easily possible to easily enumerate, identify, and conveniently obtain the full Support Information Set. It may also in some cases make sense to make targeted subsets of the Support Information Set easily identifiable and downloadable.

The ability to study, work, and otherwise operate offline as much as possible is hugely important. Unfortunately it's significantly undervalued by a disturbingly large portion of people right up to the point where they find themselves needing it and not having it. FOSS projects requiring users and/or contributors to have Internet access more than is strictly necessary are absolutely unacceptable. It is crucial that FOSS projects both encourage a taste for as well as enable self sufficiency to the greatest degree possible. True technological freedom means among other things not being a dog on the leash of one's Internet provider.

This document's primary author lives in New York City which is a relatively prosperous and technology heavy corner of the world. Nonetheless while he usually has access to high speed Internet or at least some kind of Internet the scenarios where he doesn't have it but could use it occur often enough to be noticeable, annoying, and a cause of lost

productivity in both research and work.

Furthermore the primary author is both friends with and acquainted with several people who recently/currently did/do not have Internet access at their place of residence or at best have spotty access. In many of these cases these people simply can't afford it. In other cases while they can afford it they are still on a sufficiently tight budget that the expense of buying Internet access was not justified as they could access it often enough not to be totally crippled in their life at their local library, in some cases through their local university, in some cases via the extortionate data plans for their mobile phones, and sometimes due to the generosity of neighbors who let them borrow their WiFi. This is true even now in 2023 and was even more true before the coronavirus pandemic(especially in late 2016 when work on this FRTD first started).

Some people the primary author knows even go so far as to deliberately not buy Internet access even though they can easily afford it because they find it to be too much of a distraction in their life and that it gets in the way of both productivity and family time. Yet even they have sometimes been significantly inconvenienced by not being able to download certain documentation that would have been quite useful for informational, educational, productivity, leisure, and/or entertainment purposes. The way FOSS projects inadvertently punish such people for making a rational and healthy lifestyle choice to not always be connected is unacceptable.

Another important point to keep in mind is security. Some friends of the primary author (as well as to a lesser extent the primary author as well at times) have made a point of not being connected to the Internet more than is strictly necessary. It is well known that exposure to a network especially one as big and potentially dangerous as the Internet significantly increases the chances of compromise. Any FOSS project that forces people to be connected to the Internet for it's study/use more than is necessary encourages them to potentially jeopardize their security and essentially punishes them for trying to stay safe. This is unacceptable. We won't even start on the kind of havoc this wreaks on people trying to get important work done whose threat model warrants using air gapped computers totally disconnected from any network. This has been a real issue for a scientific

software consultancy the primary author interviewed with where people working on sensitive projects temporarily had to leave a secure room/facility, go look up some Support Information of FOSS tools they were using, go back to the secure room, and then try to use it from memory. This also undoubtably is an issue for human rights activists and organizations trying to leverage technology to counteract the force/power amplification that technology also gives to oppressors and level they playing field somewhat.

Also forget security for a moment. Normal regular people definitely need offline documentation.

Of course the key thing to keep in mind is that this is the kind of stuff that's happening in one of the more prosperous regions of the world even in the year 2023. In many other places these problems are exacerbated by many orders of magnitude. In many less prosperous places Internet access is either notoriously unreliable or (almost) nonexistent. Ultimately FOSS projects that that force users to constantly need to be online to use or study them inadvertently perpetuate and increase inequality almost as much as some actively malicious predatory monopolistic interests. The rich will be able to study, learn, get better educated, get more skills, build better infrastructure, get better incomes, and get yet more disproportionately rich both with regards to money and overall quality of life. The poor and less fortunate remain largely trapped in their cycle of misery and desperation. They never get a meaningful chance to lift themselves up from that situation and into a better, more prosperous, and happier life. Both advantages and disadvantages of these kinds generally tend to compound exponentially in feedback loops the latter of which we must avoid. Let's not mince words here. Any FOSS project which does not enable self sufficiency to the largest extent reasonably possible is a big part of the problem.

Instead of forcing people to have to be lucky all the time with regards to a reliable Internet connection any halfway decent FOSS project would make it so that if someone gets lucky just once they can grab a comprehensive Support Information Set along with any other desired parts and then be able to work self sufficiently regardless of future network access. They would also be able to help their friends and people in their community by making

copies for them. If they're feeling more entrepreneurial perhaps they can even sell copies. This is something they may also need to do just to break even depending on the exact distribution mode to offset the cost of the distribution media. FOSS projects really could do worse than enabling self sufficiency, more generosity, more trade, and accelerated development for people especially in less prosperous communities. There's a world of difference between people having to get lucky once and having to be lucky all the time. (Of course ideally people wouldn't even have to get lucky once but that's impossible.)

Furthermore even in the more economically prosperous parts of the world unnecessary dependence on an Internet connection is very unhealthy both for individuals and communities. The Internet for the most part has a rather tree like and hierarchic structure which has several unpleasant ramifications.

Even in the world's less oppressive countries there exist Internet kill switches. These have rarely been used because the regimes of these less oppressive countries often find the Internet to be yet another useful medium via which to propagandize, disinform/misinform, and spy. Another reason is regimes understand that some limited freedoms including but not limited to a relatively uncontrolled Internet often make for significantly more productivity and commerce among their populations resulting in significantly more taxable revenue. Finally because of it's hierarchic nature most Internet traffic has to pass through certain centralized exchange points where it can potentially be spied on or otherwise manipulated/censored. Yet more devastatingly these centralized points can be shut down by a regime if the cost-benefit dynamic of keeping the Internet working changes. One such scenario might be people are using it to organize en masse in an attempt to address legitimate grievances and no amount of easy spying enabled by the Internet's hierarchic topological structure can contain/disrupt the movement as would be possible were it still small. Or it could be something much more mundane like misguided attempts by many countries trying to stop cheating on the national exam day.

Perhaps worst of all even if country A's regime is relatively benevolent and doesn't want to shut down the Internet there for malicious reasons the fact is

country B's regime does and perhaps one day will. Just about all countries have enemies whether open or secretive. Needless to say the centralized exchange points at or near the root of the tree in any given country make an extremely appealing target for (the intelligence agencies of) enemy countries and can safely be assumed to have been compromised covertly or otherwise by one or more of them. One day some country may very well shut down some other's Internet and with it a significant part of the critical infrastructure the latter depends on resulting in significant devastation. Of course we're getting way ahead of ourselves here. A totally random natural disaster can achieve similar effects and a disturbingly large portion of any given country's critical infrastructure isn't even ready to handle that.

You dear reader(s) do not want to be relying on the Internet more than is strictly necessary. We believe in a global network transcending all borders that people can use to collaborate, exchange ideas, solve pressing global problems, and organize mutual aid if they wish. The world needs such a network and needs it the millennium before last at the absolute latest. Unfortunately that network is not the Internet in it's present form and likely won't be the Internet in any of its future forms either though we desperately hope we're wrong about that. Even in the event a global mesh network of the same scale and span as the Internet were to exist because say Hyperboria, IPFS, Scuttlebutt, or something like them just randomly caught on and spread like wildfire we still do not want to be dependent on that network more than is strictly necessary. We deeply value cooperation and a healthy interdependence for the well being and prosperity of all people. However we also deeply value self sufficiency and independence. In most regards these two possibilities are not mutually exclusive.

Another thing worth noting is that many parts of the world don't even have reliable electricity let alone Internet access. What this implies is that the Support Information Set that comes with an FRT should try to optimize for that. For example in some rare cases where an educational video really is by far obviously the best way for an FRT implementor to get some information across by all means do that. But in most cases it really isn't and precludes people from doing things like printing out all of some targeted subset of the Support Information Set at a library or somewhere similar to enable undisrupted study.

It's totally fine and even great to choose to rely on each other for various things. It makes everyone's life easier and bolsters everyone's quality of life. The key word here however is choose. What's not so great is to have to rely on each other. Sure there are many times in life doing so is absolutely and utterly unavoidable. Nonetheless there are just as many times when it absolutely is and forming such unncessary dependences results in fragile unhealthy individuals, communities, and relationships.

We cannot stress enough that understandability via complete Open Knowledge Sets and self sufficiency are key to true technological freedom.

core requirements of the FRTs in addition to those defined by FOSS

1. The full Open Knowledge Set must be available for easy self sufficient offline download, access, and use. It must be obtainable in one go/download for any given release of an FRT.

2. There must be an Open Knowledge Set Enumeration that documents everything available in the Open Knowledge Set so that people can reason about the parts of it they may or may not already have and the parts of it they may or may not want.

3. The Open Knowledge Set Enumeration must be easily and prominently visible/discoverable at the FRT's Point Of Initial Discovery which will usually be something like a FOSS project's home page on the web where an interested party lands after first learning of the FRT's existence by some means.

4. For example in the case of software which is the primary author's area of expertise the full Open Knowledge Set includes at least built executables, program sources, the full built existing Support Information Set, and any source materials from which the Support Information Set has been derived.

5. Again the built Support Information Set must be easily downloadable for offline use in it's entirety including both Ramp Up and Reference

materials. If it can be shown/viewed online without the an outsider needing to run a build on the Support Information Set sources it can be just as easily available offline to them too.

6. Sometimes the source material for the Support Information Set of a FOSS project isn't available. But even when it is (which happily is somewhat often) one shouldn't have to deal with the hassle of setting up and running the documentation build/generator tool and it's whole recursive tree of dependencies just to build/get the Support Information Set. While that's an important skill which we absolutely encourage everyone to have it's not one anyone should need to use when dealing with FRTs.

7. We've seen many cases where we not only have to build the Support Information Set from source material but then the built Support Information Set wound up being a static set of HTML, CSS, and JavaScript files. Disturbingly often it then wouldn't work properly or at all unless we ran a web server locally. Avoid this. We've seen plenty of good Support Information Sets consisting of exactly this composition that don't require running a local web (or any other) server to read them. We could just go and open index.html in the offline documentation and study.

8. If an FRTs offline Support Information set is an offline website clone with lots of HTML files and some JavaScript the FRT should throw in some fully offline search functionality. The tooling to do so is out there and it was so beautiful to behold where we've seen it done.

9. There may be some edge cases where the Support Information Set is something truly dynamic because that's by far the best way to communicate certain knowledge. Odds are overwhelming though that your FOSS project isn't one of them.

10. Highly dynamic (aspects/subsets of) Support Information Sets like for example Jupyter style notebooks requiring a local web server or ones with lots of videos can sometimes be appropriate as they're the best way to communicate certain knowledge. But often they are not and

shouldn't be used willy nilly. Remember to enable low tech study of an FRT like for example printing out some documentation at a library. Not to mention while artificial intelligence may quite likely change this programatically searching through videos is nowhere near as easy as searching through text or better yet other forms of structured but still human friendly data. Even images while often necessary should not be overly relied on as they're inaccessible to visually impaired people using many kinds of screen reading software. Alt text and/or other such captions should be provided wherever reasonably possible as a mitigation. This also benefits the rather rare set of people operating in text only environments.

11. On the matter of internationalization a good faith attempt should be made though we understand that many projects and underresourced and can't do a perfect or even decent job of this. Perhaps recent advancements in artificial intelligence can help mitigate this. (It would be nice for there to be a universal Esperanto style language we could all agree to use and learn in addition to our native ones. If such a thing ever truly emerges Support Information Sets among other things should prioritize this before focusing on other languages.)

12. Disturbingly often the built Support Information Set just isn't available for offline study. Don't do that. It must be available for offline study. While we encourage people to be versed in using things like web crawling and web mirroring software these skills absolutely shouldn't be necessary just to study an FRT.

13. We've seen many projects where the contents of a Support Information Set as defined by the boundaries of what constitutes the project are not adequately enumerated and we can't reason about what parts of it we have nor what's even available. Don't do that. Clearly enumerate the whole Support Information Set.

14. The whole Support Information Set must be obtainable in one go/click/download for those who want just that and not the whole Open Knowledge Set purely for study purposes without say downloading and running the actual software.

15. Sufficiently large Support Information Sets should be broken into smaller useful (ideally disjoint) subsets that are easily downloadable in a targeted fashion. This way people with poor/limited Internet access and/or persistent storage space can still study the parts of the FRT they find interesting instead of being prevented from studying it entirely. This kind of breakdown may not always be possible especially in cases where there are a lot of tight interdependencies between the subsets but a good faith attempt should be made. Often this means the FRT is poorly designed but not always as the real world does sometimes have a lot of intrinsic unavoidable complexity. Similar principles apply to other parts of the full Open Knowledge Set.

16. Dependencies between any existing subsets of the Support Information Set should be specified as formally and machine readably as possible. This facilitates grabbing useful subsets via whole dependency trees both manually and in an automated fashion with various kinds of packages managers. Similar principles apply to other parts of an FRTs full Open Knowledge Set.

17. Dependencies between distinct FRTs should also be specified as formally as possible for similar reasons. Imagine being able to use a package manager pull in an FRT textbook whether by just by itself or also with the whole dependency tree of prerequisites one needs to understand it.

18. We've seen cases where the Ramp Up subset of a Support Information Set was available for offline study but the Reference Part was not even though it very much existed. Don't do that. This may enable people to get started with a technology but will prevent them from progressing if they need to look up a nontrivial detail while offline.

19. We've also seen cases where the Reference subset of a Support Information Set was available for offline use but not the Ramp Up part even though high quality ramp up material existed. This is catastrophic as Reference material is often useless if one doesn't have the conceptual background to make sense of it that the Ramp Up part

provides. This is a prime example of Open Facts instead of Open Knowledge which is exactly what we want to avoid.

20. Many FOSS projects have full Support Information Sets available for offline study but make them difficult to find unless one is really looking on their website. Don't be one of them. We are so burned out by lack of offline Support Information Sets in the majority of FOSS projects that we may simply assume they're not there and go study, use, and contribute to another project which does this.

21. There are scenarios where it's obvious that there is a complete Support Information Set (or useful subset) available for offline study but the link to it is broken. Be sure this isn't your FOSS/FRT project dear reader. To get at the Support Information Set the FOSS project intended to be as public as possible we sometimes had to use the same kind of techniques as when conducting a penetration test and searching for IDOR(Insecure Direct Object Reference) vulnerabilities. Penetration testing and hacking skills are something we strongly encourage everyone to have. But they should not be necessary even in their most basic forms to use/study an FRT.

22. We've also dealt with FOSS projects which used to have offline Support Information Sets but then no longer seemed to with newer releases. They just weren't obviously or nonobviously linked/visible anywhere on their websites. However the Support Information Set was actually there if we used IDOR vulnerability hunting type techniques to find them. We really really wish we were making this up. All the hard work FOSS projects put into assembling the Support Information Set was wasted for those users and contributors that don't have reliable Internet access, absurd amounts of patience, and penetration testing skills. Avoid this.

23. There are FOSS projects/packages where there's a full offline Support Information Set and it's installed on a system by default. However when one invokes the help functionality of that project it either a) points to the online set instead of the offline set present on the system and/or b) fails to disclose the presence of the offline set at all. To add

insult to injury sometimes these rather undiscoverable offline Support Information Sets are quite large and eat significant persistent storage space. The endlessly innovative ways in which FOSS projects manage to pull crushing defeat from the very jaws of victory itself never cease to astound us. Yet another failure mode to avoid and a small thing to change for a large impact. There needs to be an unbroken discovery chain from the main entry points of a Support Information Set to any other part of the set.

24. It should go without saying that a properly designed offline Support Information Set references the official online ones for easy discovery of Sources Of Truth so to speak.

25. Sometimes powerful help systems that then point to robust local Support Information Sets exist but it's not obvious how to invoke said help system or even that it exists. Varying interfaces have varying constraints that may or may not allow broadcasting of the help system's existence in a clearly visible yet unobtrusive fashion but a good faith attempt should be made.

26. With FRTs the most important thing for implementors to focus on is capturing as much of the knowledge they want to share as possible into the Open Knowledge Set without being overly concerned how that's distributed amongst the set's parts. That said the set's parts should be kept as uncoupled/as reasonably possible. We've seen FOSS projects where the text files making up the Support Information Set were embedded into the built executable and couldn't also be read by other text viewers/editors present on the system. Avoid coupling like this.

27. An FRTs Support Information Set must be published in at least one standardized open format (or better yet FRT compliant format). This requirement will likely be tightened in future FRTD versions as FRTs become more widespread.

28. Most real life systems including FRTs have dependencies/subsystems/subcomponents they build on. Ideally FRTs would be built from the ground up in terms of other FRTs. Such an

FRT is defined to be a Rank 1 Freedom Respecting Technology or R1FRT for short.

29. An FRT with only FOSS and FRT dependencies is defined to be a Rank 2 Freedom Respecting Technology or R2FRT for short.

30. An FRT which has any kind of dependency that's not FOSS or FRT is defined as a Rank 3 Freedom Respecting Technology or R3FRT for short.

31. When talking about FRTs it's often prudent to state what version of the FRTD they comply with to better assist people being able to reason about what kinds of freedoms the FRT affords them.

32. Suppose one implemented an adapter for the interface of a mere FOSS or totally unfree external system to make using that system more comfortable. Since the external system is not properly part of the adapter's implementation it doesn't prevent the adapter from being an FRT. Adapters should be clearly identified as such and clearly specify the external systems they are wrapping. All other build and execution time dependencies of the adapter proper must of course be FRTs.

33. Use sane default (file) names for things like projects, Support Information Sets, etc so that they can be searched for in an automated manner on one's computer. You should avoid crazy characters that cause issues in things like shell commands. We're tired of trying to figure out how to name a file holding say a Support Information Set for the TLA*\F+ project so that we actually have a chance of being able to find it later.

concluding remarks

Hopefully after having read thus far it should now be obvious that many technologies even though compliant with the various FOSS definitions still fail to provide practical technological freedom. A newer clearer vision for technological freedom was long overdue and this is our attempt to rectify

the situation.

In some cases the problems mentioned here are deliberate. Some FOSS projects are frankly just bait by predatory entities to tempt vulnerable targets. These targets could be idealistic people. These targets could be strategic thinking organizations trying to maximize their options by avoiding unfree systems locking them in only to wind up trapped in expensive commercial "support" contracts instead.

To be clear we have nothing against entities selling FOSS products and/or support for them. We all need to make a living. Our issue is with so called "open" systems not providing us the tools to help/serve ourselves when clearly they (almost) exist or could exist with some not very labor intensive modifications.

This document is not for the makers of the predatory FOSS projects mentioned above. This is for the many FOSS makers that seem to have no visible/obvious bait profit motive, that hopefully have some kind of healthy profit acquisition capabilities, that have a genuine enthusiasm for open knowledge, and that have a genuine desire to make the world a better place. It's here to make explicit the things they accidentally/thoughtlessly omit/misprioritize.

We hope that this document inspires the creation of many FRTs right from the start, independent evolution of FOSS projects into FRTs driven by their developers, and also such evolution driven by user request.

Are the requirements stated herein more work for creators? Quite possibly though many FOSS projects are much closer than they think to being FRTD compliant and only need to make minor corrections for major benefit.

Aren't implementors already often horribly badly overworked, underresourced, and sadly stuck dealing with entitled users? Very much yes. Sadly the primary author is ashamed to have been among those entitled users at points in the past.

So why then should FOSS projects aim for FRTD compliance when they

have limited resources to invest? Often when one has limited resources and a big vision they wish to execute the best use of those limited resources is conducting activities that will bring in more resources until there are at least sufficient resources for the situation at hand.

How do contributors to FOSS projects come about once they discover the project by some means? They try meaningfully studying it beyond the initial marketing fluff to see if it's suitable for their objectives whether directly or with some reasonable amount of customization work. Then once they've studied enough to meaningfully use the project and do so for a while they get a deep gut feeling of its current behavior/workings and a sense of the gap between those and the desired behavior/workings. At that point they finally know enough to contribute meaningful fixes that the project maintainers would likely find useful and incorporate. After some time if there's a consistent track record of contributions they themselves may be promoted to a maintainer if that's what they want. This is what the FOSS contribution pipeline often looks like.

Often what happens to this pipeline in non-FRT FOSS projects is that it gets disrupted very early on by newcomers not being able to meaningfully access/study the full Open Knowledge Set in a reasonably self sufficient manner let alone use it, further develop it, and contribute those developments back. FRTD compliance should be prioritized over other often less important feature work because True Open Knowledge is the meta feature that builds the community around a project and thus helps scale the resources closer to what's really needed for other development. The alternative is a death spiral of burnout once resources run out.

Open the knowledge and the builders will come.