BUGS

There are still bugs

Curl and libcurl keep being developed. Adding features and changing code means that bugs will sneak in, no matter how hard we try not to.

Of course there are lots of bugs left. And lots of misfeatures.

To help us make curl the stable and solid product we want it to be, we need bug reports and bug fixes.

Where to report

If you can't fix a bug yourself and submit a fix for it, try to report an as detailed report as possible to a curl mailing list to allow one of us to have a go at a solution. You can optionally also submit your problem in curl's bug tracking system.

Please read the rest of this document below first before doing that!

If you feel you need to ask around first, find a suitable mailing list and post your questions there.

Security bugs

If you find a bug or problem in curl or libcurl that you think has a security impact, for example a bug that can put users in danger or make them vulnerable if the bug becomes public knowledge, then please report that bug using our security development process.

Security related bugs or bugs that are suspected to have a security impact, should be reported on the curl security tracker at HackerOne.

This ensures that the report reaches the curl security team so that they first can be deal with the report away from the public to minimize the harm and impact it will have on existing users out there who might be using the vulnerable versions.

The curl project's process for handling security related issues is documented separately.

What to report

When reporting a bug, you should include all information that will help us understand what's wrong, what you expected to happen and how to repeat the bad behavior. You therefore need to tell us:

- your operating system's name and version number

- what version of curl you're using (`curl -V` is fine)

- versions of the used libraries that libcurl is built to use

- what URL you were working with (if possible), at least which protocol

  and anything and everything else you think matters. Tell us what you expected to happen, tell use what did happen, tell us how you could make it work another way. Dig around, try out, test. Then include all the tiny bits and pieces in your report. You will benefit from this yourself, as it will enable us to help you quicker and more accurately.

  Since curl deals with networks, it often helps us if you include a protocol debug dump with your bug report. The output you get by using the `-v` or `--trace` options.

  If curl crashed, causing a core dump (in unix), there is hardly any use to send that huge file to anyone of us. Unless we have an exact same system setup as you, we can't do much with it. Instead we ask you to get a stack trace and send that (much smaller) output to us instead!

  The address and how to subscribe to the mailing lists are detailed in the `MANUAL.md` file.

libcurl problems

When you've written your own application with libcurl to perform transfers, it is even more important to be specific and detailed when reporting bugs.

Tell us the libcurl version and your operating system. Tell us the name and version of all relevant sub-components like for example the SSL library you're using and what name resolving your libcurl uses. If you use SFTP or SCP, the libssh2 version is relevant etc.

Showing us a real source code example repeating your problem is the best way to get our attention and it will greatly increase our chances to understand your problem and to work on a fix (if we agree it truly is a problem).

Lots of problems that appear to be libcurl problems are actually just abuses of the libcurl API or other malfunctions in your applications. It is advised that you run your problematic program using a memory debug tool like valgrind or similar before you post memory-related or "crashing" problems to us.

Who will fix the problems

If the problems or bugs you describe are considered to be bugs, we want to have the problems fixed.

There are no developers in the curl project that are paid to work on bugs. All developers that take on reported bugs do this on a voluntary basis. We do it out of an ambition to keep curl and libcurl excellent products and out of pride.

But please do not assume that you can just lump over something to us and it will then magically be fixed after some given time. Most often we need feedback and help to understand what you've experienced and how to repeat a problem. Then we may only be able to assist YOU to debug the problem and to track down the proper fix.

We get reports from many people every month and each report can take a considerable amount of time to really go to the bottom with.

How to get a stack trace

First, you must make sure that you compile all sources with `-g` and that you don't 'strip' the final executable. Try to avoid optimizing the code as well, remove `-0`, `-02` etc from the compiler options.

Run the program until it cores.

Run your debugger on the core file, like `<debugger> curl core`. `<debugger>` should be replaced with the name of your debugger, in most cases that will be `gdb`, but `dbx` and others also occur.

When the debugger has finished loading the core file and presents you a prompt, enter `where` (without quotes) and press return.

The list that is presented is the stack trace. If everything worked, it is supposed to contain the chain of functions that were called when curl crashed. Include the stack trace with your detailed bug report. It'll help a lot.

Bugs in libcurl bindings

There will of course pop up bugs in libcurl bindings. You should then primarily approach the team that works on that particular binding and see what you can do to help them fix the problem.

If you suspect that the problem exists in the underlying libcurl, then please convert your program over to plain C and follow the steps outlined above.

Bugs in old versions

The curl project typically releases new versions every other month, and we fix several hundred bugs per year. For a huge table of releases, number of bug fixes and more, see: https://curl.se/docs/releases.html

The developers in the curl project do not have bandwidth or energy enough to maintain several

branches or to spend much time on hunting down problems in old versions when chances are we already fixed them or at least that they've changed nature and appearance in later versions.

When you experience a problem and want to report it, you really SHOULD include the version number of the curl you're using when you experience the issue. If that version number shows us that you're using an out-of-date curl, you should also try out a modern curl version to see if the problem persists or how/if it has changed in appearance.

Even if you cannot immediately upgrade your application/system to run the latest curl version, you can most often at least run a test version or experimental build or similar, to get this confirmed or not.

At times people insist that they cannot upgrade to a modern curl version, but instead they "just want the bug fixed". That's fine, just don't count on us spending many cycles on trying to identify which single commit, if that's even possible, that at some point in the past fixed the problem you're now experiencing.

Security wise, it is almost always a bad idea to lag behind the current curl versions by a lot. We keeping discovering and reporting security problems over time see you can see in this table

Bug fixing procedure

What happens on first filing

When a new issue is posted in the issue tracker or on the mailing list, the team of developers first need to see the report. Maybe they took the day off, maybe they're off in the woods hunting. Have patience. Allow at least a few days before expecting someone to have responded.

In the issue tracker you can expect that some labels will be set on the issue to help categorize it.

First response

If your issue/bug report wasn't perfect at once (and few are), chances are that someone will ask follow-up questions. Which version did you use? Which options did you use? How often does the problem occur? How can we reproduce this problem? Which protocols does it involve? Or perhaps much more specific and deep diving questions. It all depends on your specific issue.

You should then respond to these follow-up questions and provide more info about the problem, so that we can help you figure it out. Or maybe you can help us figure it out. An active back-and-forth

communication is important and the key for finding a cure and landing a fix.

Not reproducible

For problems that we can't reproduce and can't understand even after having gotten all the info we need and having studied the source code over again, are really hard to solve so then we may require further work from you who actually see or experience the problem.

Unresponsive

If the problem haven't been understood or reproduced, and there's nobody responding to follow-up questions or questions asking for clarifications or for discussing possible ways to move forward with the task, we take that as a strong suggestion that the bug is not important.

Unimportant issues will be closed as inactive sooner or later as they can't be fixed. The inactivity period (waiting for responses) should not be shorter than two weeks but may extend months.

Lack of time/interest

Bugs that are filed and are understood can unfortunately end up in the "nobody cares enough about it to work on it" category. Such bugs are

perfectly valid problems that *should* get fixed but apparently aren't. We try to mark such bugs as `KNOWN_BUGS material` after a time of inactivity and if no activity is noticed after yet some time those bugs are added to `KNOWN_BUGS` and are closed in the issue tracker.

KNOWN_BUGS

This is a list of known bugs. Bugs we know exist and that have been pointed out but that haven't yet been fixed. The reasons for why they haven't been fixed can involve anything really, but the primary reason is that nobody has considered these problems to be important enough to spend the necessary time and effort to have them fixed.

The `KNOWN_BUGS` are always up for grabs and we will always love the ones who bring one of them back to live and offers solutions to them.

The `KNOWN_BUGS` document has a sibling document known as `TODO`.

TODO

Issues that are filed or reported that aren't really bugs but more missing features or ideas for future improvements and so on are marked as 'enhancement' or 'feature-request' and will be added to the `TODO` document instead and the issue is

closed. We don't keep TODO items in the issue tracker.

The `TODO` document is full of ideas and suggestions of what we can add or fix one day. You're always encouraged and free to grab one of those items and take up a discussion with the curl development team on how that could be implemented or provided in the project so that you can work on ticking it odd that document.

If the issue is rather a bug and not a missing feature or functionality, it is listed in `KNOWN_BUGS` instead.

Closing off stalled bugs

The issue and pull request trackers only holds "active" entries open (using a non-precise definition of what active actually is, but they're at least not completely dead). Those that are abandoned or in other ways dormant will be closed and sometimes added to `TODO`and `KNOWN_BUGS` instead.

This way, we only have "active" issues open on github. Irrelevant issues and pull requests will not distract developers or casual visitors.