Frequently Asked Questions

This document answers some frequently asked questions about the Protocol Buffers open source project. If you have a question that isn't answered here, join the discussion group and ask away!

General

Why did you release protocol buffers?

There are several reasons:

- Protocol buffers are used by practically everyone inside Google. We have many other projects we would like to release as open source that use protocol buffers, so to do this, we needed to release protocol buffers first. In fact, bits of the technology have already found their way into the open – if you dig into the code for Google AppEngine, you might find some of it.
- We would like to provide public APIs that accept protocol buffers as well as XML, both because it is more efficient and because we're just going to convert that XML to protocol buffers on our end anyway.
- We thought that people outside Google might find protocol buffers useful.
- Getting protocol buffers into a form we were happy to release was a fun 20% project.

Why is the first release version 2? What happened to version 1?

The initial version of protocol buffers (aka "Proto1") was developed in Google starting in early 2001, and evolved over the course of many years, sprouting new features whenever someone needed them and was willing to do the work themselves. Like anything created in such a way, it was a bit of a mess. We came to the conclusion that it would not be feasible to release the code as it was.

Version 2 ("Proto2") is a complete rewrite, though it keeps most of the design and uses many of the implementation ideas from Proto1. Some features have been added, some removed. Most importantly, though, the code is cleaned up and does not have any dependencies on Google libraries that have not yet been open-sourced.

Why the name "Protocol Buffers"?

The name originates from the early days of the format, before we had the protocol buffer compiler to generate classes for us. At the time, there was a class called `ProtocolBuffer` which actually acted as a buffer for an individual method. Users would add tag/value pairs to this buffer individually by calling methods like `AddValue(tag, value)`. The raw bytes were stored in a buffer which could then be written out once the message had been constructed.

Since that time, the "buffers" part of the name has lost its meaning, but it is still the name we use. Today, people usually use the term "protocol message" to refer to a message in an abstract sense, "protocol buffer" to refer to a serialized copy of a message, and "protocol message object" to refer to an in-memory object representing the parsed message.

Does Google have any patents on Protocol Buffers?

Google currently has no issued patents on Protocol Buffers, and we are happy to address any concerns around Protocol Buffers and patents that people may have.

Similar Technologies

How do protocol buffers differ from XML?

See the answer on the overview page.

How do protocol buffers differ from ASN.1, COM, CORBA, Thrift, etc?

We think all of these systems have strengths and weaknesses. Google relies on protocol buffers internally and they are a vital component of our success, but that doesn't mean they are the ideal solution for every problem. You should evaluate each alternative in the context of your own project.

It is worth noting, though, that several of these technologies define both an interchange format and an RPC (remote procedure call) protocol. Protocol buffers are just an interchange format. They could easily be used for RPC – and, indeed, they do have limited support for defining RPC services – but they are not tied to any one RPC implementation or protocol.

Contributing

Can I add support for a new language to protocol buffers?

Yes! In fact, the protocol buffer compiler is designed such that it's easy to write your own compiler. Check out the `CommandLineInterface` class, which is available as part of the `libprotoc` library.

We encourage you to create code generators and runtime libraries for new languages. You should start your own, independent project for this – this way, you will have the freedom to manage your project as you see fit, and will not be held back by our release process. Please also join the Protocol Buffers discussion group and let us know about your project; we will be happy to link to it and help you out with design issues.

Can I contribute patches to protocol buffers?

Yes! Please join the Protocol Buffers discussion group and talk to us about it.

Can I add new features to protocol buffers?

Maybe. We always like suggestions, but we're very cautious about adding things. One thing we've learned over the years is that lots of people have interesting ideas for new features. Most of these features are very useful in specific cases, but if we accepted all of them, protocol buffers would become a bloated, confusing mess. So, we have to be very picky. When evaluating new features, we look for additions that are very widely useful or very simple – or hopefully both. We regularly turn down feature additions from Google employees. We even regularly turn down feature additions from our own team members.

That said, we'd still like to hear what you have in mind. Join the Protocol Buffers discussion group and let us know. We might be able to help you find a way to do what you want without changing the underlying library. Or, maybe we'll decide that your feature is so useful or so simple that it should be added.

Last updated 四月 *22, 2014.*