
Standards and specs: Naturally occurring standards

When standards form in the wild, it may be a cry for formal standardization

Skill Level: Introductory

[Peter Seebach \(crankyuser@seebs.plethora.net\)](mailto:crankyuser@seebs.plethora.net)

Author

Freelance

12 Apr 2005

What makes a standard viable without the formal blessing of a standards organization? Should you use such informal standards, or ignore them? Learn more about de facto standards in this month's Standards and specs.

Dave Clark once said, "We reject kings, presidents and voting. We believe in rough consensus and running code."

The IETF (the organization Dave Clark was speaking for) is a standards body that has taken an unusual, but empirically quite effective, approach to standardization, preferring to see proposed standards tested out a bit in the field before they get formal blessing. In short, whenever possible, they formalize an existing de facto standard, rather than inventing a new one from scratch. The RFC process (see [Resources](#)) has worked well enough to produce many of the most widely used standards in the world, and IETF standards have a credibility even ISO can't always match.

What makes a de facto standard good enough to formalize, or possibly so good it doesn't even need to be formalized? In the wild, you will often encounter standards, with or without the blessings of a standards body, which seem like they might be applicable to your work. Sometimes, you may find no applicable standard, but a likely partnership to create one. In this Standards and specs, you'll see a few things to keep in mind when talking about de facto standards.

First, though, to dispel a few myths: Not all de facto standards are the same. Some

of them are really good. Some are really bad. Not every de facto standard represents the best possible technical decisions; not every de facto standard represents the tyranny of a proprietary despot dribbling out just enough crumbs of documentation to keep the peasants from revolting. De facto standards can be temporary kluges, or carefully considered and planned designs; they can reflect an individual's vision or a committee's indecision. In short, it is dangerous to treat them as interchangeable.

Physical connectors

Computers love to talk to other devices. This (unless you're using radio) requires them to have some kind of cable connecting them, and that means connectors for the cable to plug into.

In most cases, a connector intended to be shared or used by multiple devices will be the topic of a formal standard. There is ample documentation on the pinouts of a standard USB cable, for instance. However, if it isn't the topic of a formal standard, the compelling need for interoperability is likely to lead to an informal standard. Connectors with formal standards are generally fairly precisely defined. The RS232 specification for serial ports describes signals, voltage levels, and everything, for standard DB25 serial ports. However, many computers ship with a 9-pin serial port. The pinouts for these have become a de facto standard.

It's worth noting that the formal standard (RS232) solved the hard problem, and the informal standard just covered a common extension. Network effects make standards like this a pretty good bet. If you're making serial cables with 9-pin connectors, they need to plug into the "standard" PC pinout. Given the widespread availability of such cables, if you're making a serial port, a 9-pin male serial port with the same pinout is a good bet. This is a great standard: it's free of licensing requirements, it's very easy to implement, and millions of machines use it. Given this, it seems obvious that anyone looking to build a serial port would want to use this standard, and certainly, anyone using a 9-pin d-sub connector would use it.

This makes it frankly inexplicable that I have a few uninterruptible power supplies that use a 9-pin d-sub connector, which is connected by a cable to a PC serial port, but which has some other pinout, such that the special serial cable the UPS comes with is the only one that can be used with it, and that cable cannot be used with other serial devices. This is a recurring theme, and it is one of the weaknesses of de facto standards: people who ought to know better sometimes ignore them. Since there's no licensing, there's also no authority to appeal to that will prevent vendors from doing this.

More de facto serial port standards

Unfortunately, not quite all serial ports are obediently standardized, and the additional complication of DTE/DCE devices leaves room for further improvement. Even with only standard ports, any given device could have a male or female connector, DTE or DCE signaling, and use any of several connector form factors. One likely solution is to develop a standard port and universal cable. Dave Yost developed a spec for a universal standard serial port (see [Resources](#)) using an RJ45 connector. Each port gets an adapter bolted onto it, and after that, all ports are interchangeable. All you need for a serial cable between any two machines is an 8-wire RJ45 cable which connects pin 1 to pin 8, pin 2 to pin 7, and so on. This is a technically excellent spec, and freely available. If you use serial ports, you owe it to yourself to convert to it. It will make your life easier.

Ironically, some vendors have thwarted this budding standard by introducing products using RJ45 ports for serial cables -- with a different pinout. What were they thinking? It's quite possible they simply weren't aware of the existing standard. The take-home lesson is to seek out existing standards before inventing your own.

In general, a de facto standard for a physical connector is a great deal better than no standard at all. Resist the temptation to improve on the standard, especially if the improvements might be subtle or confusing.

Standards for connectors tend to stay fixed for a very long time simply because physically compatible plugs which are not electrically compatible make users come after manufacturers with torches and pitchforks. There's really no middle ground when building a connector -- either make it compatible with other connectors or make it physically distinct from them. Don't give the user an opportunity to cause damage due to design neglect.

For a spectacular example of a company getting this wrong, consider this example -- a piece of music gear that uses a 5-pin DIN connector for power connection. If you have 30 5-pin DIN connectors on the back of a rack of equipment and 28 of them are MIDI connectors, one is a comparatively high-voltage power source, and one needs that high-voltage power connected to it -- that's not so good. A quick guideline: If you feel the need to resort to 30-point bold text in your product manual warning people of the possibility of destroying hardware if they do something that would be intuitively obvious, something went wrong. Go back to the drawing board. Do not pass Go, do not collect two hundred dollars. Just pick a physically distinct connector. The world is full of connectors.

De facto 2: Attack of the clones

If a product is successful, imitators will arise. A technically compatible imitation of a product creates, in effect, a primitive de facto standard. A clone standard is generally not a first choice -- its strengths will come largely from network effects. For instance, the clones of the original IBM® PC have left us with machines with 3Ghz processors and 4GB of RAM which still distinguish between the 640K of "base" memory and the almost-4GB of "extended" memory. Ouch. And yet, obnoxious though this may be, it's in many ways superior to the mass of subtly incompatible systems it replaced. The history of the term "PC compatible" is long and interesting; see [Resources](#) for more about it.

Clone standards suffer greatly from a lack of direction and design. The original design may have been fairly good, but unless the clones are all perfect, the net result will be a standard full of landmines. For instance, a number of vendors cloned DEC's Tulip ethernet chip. The original chip was a technically excellent chip, very friendly to driver writers. The mass of clones is such that NetBSD's driver for the family of chips now lists 25 distinct chipset types, with an amazing variety of quirks to take into account.

A somewhat more venerable standard based on a cloned chip is the 16550 UART controller chip. This chip has been cloned by so many companies, in so many ways, as to beggar the imagination. Nearly every operating system has drivers for this piece of hardware -- Mac OS X supports it even though I can't find a consumer Macintosh product that has ever had one! The need for support for serial ports in server machines, though, has left them with a good reason to have a driver for the hardware. This simple piece of hardware, originally made by National Semiconductor, has survived several bus architectures. It was first seen in the early days of ISA and is still the standard serial interface on modern PCI systems. The interface is so well-documented and straightforward that there are an unbelievable number of clones.

Now that all the operating systems support this hardware, it would be crazy for anyone developing a new serial chip not to implement its interface. This shows network effects going wild on a technically reasonable standard. There's no big-name consortium backing this standard. It's just well-documented and easy to implement.

So, even with their flaws, clone standards can be beneficial. Running NetBSD on an old version of the VirtualPC PC emulator required only a tiny amount of driver work, because the emulated ethernet controller was Tulip-compatible. Mostly, anyway. Connectix benefitted from widespread OS support, because it used a de facto standard. Systems which supported this de facto standard likewise benefitted from access to the platform.

The biggest danger of clone standards is probably the comparative lack of

documentation, especially in boundary conditions. Some clone standards have serious technical flaws. More subtly, a clone standard may include patented technology; discovering -- probably too late -- that your product depends on patented technology can wreck your day.

Let my people go

In practice, a word processor that can't read Microsoft® Word documents is an economic dead end. The formats used by the Microsoft Office applications have become a de facto standard, giving Microsoft a substantial competitive edge because each *new* release of its software can deliver for it a window of opportunity during which only its software is *fully* compatible; this is mitigated a bit, though, because incompatibility in a new version makes customers slow to upgrade to that newest version.

This is the evil twin brother of the cloned standard: a proprietary standard. Proprietary standards are generally bad for everyone but the vendor who controls them. They are only really seen as standards when the network effect makes them so ubiquitous that it's economically insane not to try to comply with the standard.

This is an area where a more formal standardization process would be beneficial to users, but there is a strong economic incentive for the competition in the field not to cooperate too much. The market leader has dominance already and won't benefit from helping competitors steal customers away. The market leader controls the standard, and will benefit most from squelching standards efforts. In fact, large competitors may also benefit from the lack of a standard. No word processor supports documents from other word processors flawlessly, so I own word processors sold by Microsoft, Corel, Sun®, and Apple, and I actually have to use all four of these to read the documents people send me! In the short run, only users would benefit from standardization.

If you're considering getting into a field where there's a proprietary de facto standard, be aware that the vendor who controls that standard has no economic reason to make it easy for you to participate. You will face poor or inaccurate documentation, if any, and quite possibly software licensing agreements prohibiting reverse-engineering. Worse, the myriad potential incompatibilities you might introduce will annoy users. The only reason to play this game is that users demand the ability to deal with files, data, or documents in proprietary formats. Otherwise, stay far away from the proprietary formats. It's not just office software; EDA tools have similar issues.

If you're a new player in a market like this, a good open format can be a selling point. Support the existing formats, but make your own open, and market pressure may

give you an excellent opportunity to make inroads. You may rest assured that users are not currently thinking to themselves, "I want the next product I buy to lock me in to a proprietary format forever".

Licensing and patents

In some cases, a standard comes with some kind of licensing restrictions, or involves something that someone has a patent on. For instance, Unisys had a patent governing a bit of the algorithm used for GIF images. In general, patents are a huge weakness for a standard. The MP3 standard is used very widely by people who simply don't know -- or don't care -- that someone theoretically has a patent on part of it, and only some code using the patented algorithm actually has a license from the patent holder. Developers and users can be bitten by this many years after they make the design decision to use a patented algorithm, due to the nature of patents (see [Resources](#)). De jure standards often require contributors to clearly disclose any known patents; de facto standards generally have no way to do this.

In the case of GIF, GIF images, while a formalized standard, enjoyed substantial popularity as a de facto standard for Web graphics. It was fallout from the patent issue that led a number of people to actively pursue adoption of the technically superior PNG format. The network effects that made GIF so hard to escape were eventually overcome by a barrier to usage. Once PNG made it into major browsers, it became a more viable standard.

Licensing can help establish a standard. UNIX® can trace its widespread adoption back to the willingness of AT&T to license the system to people. The GNU products have in some cases been able to become de facto standards simply by being widely available and free. Their licensing terms are very clear and are furthermore economically excellent for most users. The IBM licensing of Power Architecture™ results in other vendors selling compatible CPUs in markets that IBM isn't selling in, producing wider adoption. (See [Resources](#) for more about the IBM interest in making Power Architecture a standard.)

Licensing tends to replace the technical problems associated with clones of proprietary products with economic problems. In some cases, it is the licensor who suffers the most from the standard. When Apple licensed software to other vendors, allowing them to create Macintosh-compatible computers, the result was a huge chunk taken out of its own market share. Of course, licensees can suffer too. When Apple discontinued the licensing program, the companies that had been selling Macintosh clones mostly disappeared.

If you're looking at licensing something, check the terms of the license carefully. If you can get comparable results without licensing issues, do so. However, network

effects may decide the issue -- no matter how free Ogg Vorbis is, car stereos are more likely to play MP3 files.

Extensions

In some cases, a de facto standard builds on a more formal standard published by a standards body. For instance, in the early days after Netscape's release, many Web pages were developed either to take advantage of special features of Netscape or to work around bugs in it. The trend continues today since many sites render as intended only in one or two browsers. The W3C's formal standard has been trumped by the practical reality of making Web sites which work in at least some browsers.

In many cases, browsers have mutually exclusive bugs so a single page can't render correctly in all of them. This becomes more noticeable as pages use more and more complicated features.

In principle, the W3C's standards suggest a solution to this known as *graceful degradation*, in which a browser that is unaware of a tag should display the page reasonably without knowing about it. For example, the `DIV` tag in modern HTML, which gives layout hints to the browser, can be simply ignored and result in a perfectly reasonable page. By contrast, the implementation of the `FRAMESET` tag has left quite literally millions of pages which are indexed in search engines as containing only the text "This page uses frames, but your browser does not support them."

From an economic standpoint, it's hard to blame people for developing for the market reality instead of the theoretical specification, but the long-term effects can be incredibly expensive and crippling. A Web page trying to do a very simple thing may end up with 40KB of very-hard-to-understand JavaScript (or should I say ECMAScript?) which tries to correctly do that simple thing in a different way for each of fifteen subtly varied combinations of browser and platform.

Standards like this can be pernicious for developers and users alike when the boundaries between the formal standard and the extensions are blurred and unclear. For instance, shell scripts written for Linux™ systems often only work in `bash`. Likewise, many Linux programs will only compile with the GNU C compiler. These bugs can be nightmarish to track.

If you're thinking about depending on a fairly common extension to a formal standard, consider the effects on your audience. Careless statistical analysis can burn you here. A standard with 90% adoption according to a survey may have much lower adoption among your target market. Keep in mind the notion of graceful degradation.

There are serious risks with de facto extensions. Often, their subject matter is eventually adopted as a de jure standard, but the final standard may be subtly different from the original implementation. The battle between Netscape and IE showed that extensions can quickly turn into a morass of proprietary extensions. Worse, there is a real risk of critical technical deficiencies: extensions are often kluges rather than carefully considered designs. The "embrace and extend" philosophy offers developers the worst of both worlds: the anti-competitive practices of a proprietary standard coupled with the design vision of a poorly-run committee.

In general, stay clear if you can. There are exceptions -- both the 9-pin serial port and Dave Yost's serial standard are extensions, but neither is proprietary -- but in many cases, a de facto standard of this sort ends up being a dead end. Be especially careful to check for possible patents on standards like this (see [Resources](#) for more on patent issues).

Roll your own

You may find yourself doing something for which there is no existing standard, formal or otherwise. A little thought and design early on can make your product a de facto standard and encourage adoption. This is generally a good thing. It can increase the pool of products you interoperate with, and it can buy you credibility and respect.

One thing to consider early on is whether there are other companies or products you'd like to interoperate with, or whom you'd like to see adopt your proposed standard. If there are, it may be worth talking with them to get some second opinions. While people love to badmouth committees, a handful of technical people working together can accomplish a great deal, and a good review of a design can make the difference between a barely-usable niche design and a workable standard.

If you want your design to become a standard, two words mean more than anything else: good documentation. You need good, clear, documentation that covers everything people need to know to implement your standard, and you need it to be easily available. Don't think of this as a trade secret. Think of it as the published interface to encourage other people to use your product. The documentation should be as complete as you can manage. Be sure to solicit feedback. If people ask questions, make sure the documentation answers those questions.

Don't invent a new standard just because you don't much care for an existing one. The costs of competing standards are huge, for both users and developers. For a competing standard to be worth it, the technological advantages have to be so great, and so clear, that no rational designer would ever use the old standard once they knew about yours. This hurdle is higher than most people think. Consider that Open

Firmware hasn't yet replaced the old PC BIOS on x86 hardware.

If you're cooperating with others on a standard, make sure that any intellectual property involved is available on reasonable terms -- ideally, completely and totally unrestricted. A standard that imposes unacceptable terms will be ignored. Hiding a patent and springing it on people later is just plain evil. Wise users will destroy any company doing such a thing, and salt the earth where the corporate headquarters once stood. The damage to everyone, developer and user alike, is simply unacceptable.

Summary

De facto standards can be anything from a considered technical effort by people with an interest in open communication (as in the case of many of the standards later blessed by the IETF), to a proprietary format, reverse-engineered out of simple economic necessity, to a solution put forth by a single person (such as Dave Yost). Standards are pursued because they offer significant benefits to developers, even when they impose substantial costs.

A good standard can reduce development time, offer economies of scale, and offer interoperability with other products. All of these benefits are possible whether the standard has been formalized by a standards organization or not. Even when it is possible to offer technical improvements to a standard, it may be more cost-effective to use the standard. It doesn't need to be perfect, and it doesn't need to be blessed by an institution, it just needs to be better than complete anarchy.

Many people have very strong opinions about standardization. Some argue that a standard that doesn't evolve in the wild can't possibly be technically sound, because committees are invariably governed more by politics than by technical considerations. Others claim that de facto standards are inherently unreliable or untrustworthy. In practice, de facto standards are neither the silver bullet which will cure all ills, nor a horrible blight on the competitive landscape.

One of the largest advantages of de jure standards is comparatively open disclosure of intellectual property issues. Unfortunately, "sleeper" patents can result in a third party having legal rights to material invented independently by others; there is no guarantee of safety from such claims. Nonetheless, a de jure standard will generally offer better disclosure.

When evaluating a standard, consider its openness, technical merit, and installed base. If there is any way to avoid closed standards, or standards which impose licensing costs, do so. A properly documented standard is dramatically preferable to having to reverse-engineer something yourself. If all else fails, consider making a

standard -- but make sure you make a standard you would have liked to use. Open, well-documented, and technically coherent standards are valuable to everyone.

About the author

Peter Seebach

Peter Seebach is trying to become a de facto standard in freelance writers, despite a snafu in the ISO standardization process. He is available in both metric and imperial sizes.