PGI Requirements 153 Clarification
Mark Morgan

The original PGI Requirement 153 text is:

> "Backwards compatibility (on-the-wire, e.g. SOAP Level, new portType, etc.) is not MUST, e.g. The requirement is that we don't necessarily must be backwards compatible but we can if we want."

I'll start clarifying this requirement by re-wording it as follows:

> "It is not required that we maintain backwards compatibility with existing specifications and port types. However, it is also not required that we explicitly break or ignore such existing specifications. In other words, compatibility and usage of existing specifications is neither required nor prohibited"

Having restated the original requirement in what I hope is a clearer way, I feel obliged to both disagree with this requirement, and to point out that from a technical point of view, this isn't even a proper requirement. From a strict point of view, a requirement is something that you are required to do (or in the limit not do). Saying that you can do something, or that you can do the opposite is the antithesis of a requirement and does not belong in a requirements document. Further, even if we assume that this is a requirement, I have to disagree on principle with the statement itself. Reuse of existing code, specification, etc. is a cornerstone of software development methods and is in the best practices of the field. Not only does it appropriately acknowledge and utilize the endeavors and effort of those who came before us, but it also encourages interoperability and interchangeability by decreasing the set of possible implementations existing for a given "function". Consider this example. Assume that there are n known non-interoperable specifications ($\mathbf{S} : \{ s_0, s_1, \ldots s_{n-1} \}$) for some grid function **F**. Now, randomly choose two implementations of function **F** (assuming a uniform distribution of implementations of **F** over **S**). Then, the chances of these two implementations being interoperable would be $1 / n$ (disregarding weirdness such as the "Tuesday Birthday Problem"[1] which I find admittedly confusing, disturbing, and delightful all at the same time). Given this, it is clear that reducing n increases the chances of interoperability, and increasing n decreases those chances. In my opinion, the only

[1] The "Tuesday Birthday Problem" is a delightfully confusing conundrum of statistics where seemingly irrelevant data information affects the probability of some occurrence. I found this article on it very interesting http://sciencenews.org/view/generic/id/60598/title/When_intuition_and_math_probably_look_wrong.

valid reason to create a new specification for a function for which a specification already exists is if one determines that existing specifications are flawed in some fundamental way (note that this does not include specifications which are merely insufficient as those can be remedied by extension).