Subject: Re: Value: why not float support?
Posted by kohait00 on Tue, 28 Dec 2010 12:47:47 GMT
View Forum Message <> Reply to Message

naah...that was not what i meant
anyway, permit another question..

why does Value support int? int64 would do it just the same?
why bool? it's actually an int or even int64 for the cpu / compiler.

so breaking this down, the only meaningfull types would be double and int64, as representing the superset of possible types. which i suppose would be the perfect case.. but in terms of handling it'd be a nightmare, a lot of (int)myval, (bool)myval or myval = (double)myfloat and (int64)myint. why not have Value do all this mess?

if the user decides to represent it's data as double, or as float, or int or int64 he also decides on the resolution he needs, well knowing that this might 'implicitly' reduce some resolution, if coming from double precision i.e..

sorry to bother you with all that. i clearly can understand that you are not well with the thought of extending / breaking a working code just for the benefit of some marginal usecases. i'd do the same. mi point is, that i see a lot more potential in usage of Value than is currently possible. this is genious class that eases handling with types a LOT. but it needs some leverage of burdens.

just to demonstrate: i had to program a typed interface for a gui in our enterprise, and, just because i was 'scared' of Value, knowing it was not able to easy deal with float (because this is what our communication protocol uses the most) left it out to be used. instead, i took a templated approach, inveneted a custom database for objects, etc. etc.. now looking back, having better understood Value and having float support, i'd save me a LOOOT af work and headache.

generally, i'd recommend to enrich Value to support all types which differ in sizeof(), these are the signed variants.. this makes Value really attractive in usage in communication protocols.

bool
char
short
int
int64
float
double