Subject: Re: FIX: UPP::Scan returns int64 for INT_V
Posted by kohait00 on Tue, 17 May 2011 09:08:25 GMT
View Forum Message <> Reply to Message

it's actually no problem in python. i do the translation as expected. int Value -> integer, int64 Value -> Long (which is not correct as well, python long: unlimited digits). this works fine. the problem arises for ConvertInt.

as i now understand ConvertInt, it's offering a margin check and returns an ErrorValue if the scanned int64 surpasses the int margin, so it's an error check feature. (the margins are the only difference between ConvertInt and ConvertInt64). ConvertInt expects you not to assume the type (int, int64, bool, actually there is not ConvertBool), it just garantees you get *some* integer value matching the expected margins of an 'int' which implies the usage of a data type as well. so i still dont quite see a problem in returning an int after error check.

since ConvertInt is used quite often (anytime scanning a String), it always generates a long in python, printing it results in '123L'.. i.e if one sets up an EditInt and expects in python to have an integer and not a long. which is not a problem, python deals with it just fine, but it's not clean. though python does the conversions as well as upp, it's still aware of the type differences
http://docs.python.org/reference/datamodel.html#the-standard -type-hierarchy

i'd really vote for doing it as in python
Quote:
Plain integers
These represent numbers in the range -2147483648 through 2147483647. (The range may be larger on machines with a larger natural word size, but not smaller.) When the result of an operation would fall outside this range, the result is normally returned as a long integer

so a ConvertInt would return the smallest representation possible *after* checking margins.

BTW: have you already checked out the BoostPyTest?