Subject: Value<int64> and Value<int> mess
Posted by Mindtraveller on Fri, 21 May 2010 08:36:54 GMT
View Forum Message <> Reply to Message

It looks like U++ int and int64 inside Value make mess.
In great number of cases, creating Value with int inside makes in Value<int64> instead. Which leads to assertion break while trying to process this Value. This is painfully and takes much time to debug.
I failed to find any conscious rules when int64 is generated instead of int. Take a look at this example:
int h = ValueTo<int>(StdConvertInt().Scan(ts.Left (delim)));gives assertion error, because actually Value<int64> is generated.

Subject: Re: Value<int64> and Value<int> mess
Posted by mirek on Fri, 21 May 2010 14:08:02 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Fri, 21 May 2010 04:36It looks like U++ int and int64 inside Value make mess.
In great number of cases, creating Value with int inside makes in Value<int64> instead. Which leads to assertion break while trying to process this Value. This is painfully and takes much time to debug.
I failed to find any conscious rules when int64 is generated instead of int. Take a look at this example:
int h = ValueTo<int>(StdConvertInt().Scan(ts.Left (delim)));gives assertion error, because actually Value<int64> is generated.

Before I start investigating, can you make my life easier and post what is in "delim" ?

(Other than that, all numeric Values - bool, int, int64, doubl - are interconvertible. So it should not really matter what is the source type).

Mirek

Subject: Re: Value<int64> and Value<int> mess
Posted by Mindtraveller on Fri, 21 May 2010 16:22:20 GMT
View Forum Message <> Reply to Message

I'm afraid these types are not interconvertible in my practice (ValueTo<> generates exception in many cases).
Here is simplified version of previous code:

```
CONSOLE_APP_MAIN
{
 String s = "11";
 int i = ValueTo<int>(StdConvertInt().Scan(s)); //generates exception
```

}

while this code works:
CONSOLE_APP_MAIN
{
 String s = "11";
 int i = ValueTo<int64>(StdConvertInt().Scan(s)); //OK!
}

It is just one of many cases (the absence of interconvertibility makes handling Ctrl::GetData() a headache too).

Subject: Re: Value<int64> and Value<int> mess
Posted by mirek on Wed, 26 May 2010 11:41:17 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Fri, 21 May 2010 12:22I'm afraid these types are not interconvertible in my practice (ValueTo<> generates exception in many cases).
Here is simplified version of previous code:
CONSOLE_APP_MAIN
{
 String s = "11";
 int i = ValueTo<int>(StdConvertInt().Scan(s)); //generates exception
}

while this code works:
CONSOLE_APP_MAIN
{
 String s = "11";
 int i = ValueTo<int64>(StdConvertInt().Scan(s)); //OK!
}

It is just one of many cases (the absence of interconvertibility makes handling Ctrl::GetData() a headache too).

It is because you are making your life hard using ValueTo

Try this:

CONSOLE_APP_MAIN
{
 String s = "11";
 int i = StdConvertInt().Scan(s);
}

ValueTo is supposed to be used in two cases:

- as part of implementation of RichValue type (type with "full support" - hashcode, equality comparison, direct coversion)

- as a way to extract RawValue type (usually some of your type used to implement something, when you do not bother about RichValue traits, just simply need to pass such type as Value through).

Sorry for incomplete docs. In fact, Value is one of last places without proper documentation - unfortunately it is also one that quite hard to document....
