Subject: ASSERT when using ValueMap
Posted by NilaT on Tue, 21 Feb 2017 15:23:04 GMT
View Forum Message <> Reply to Message

Hello everyone, I've got a nice little "soft crash" again, may you can help me?
It happens when I try to "Add" something to a Value Map.
So my code basically looks like this:

Params params = HandleParams(param1, param2); // Params is a struct which contains some Strings, some ints, and a Time Variable
ValueMap result;
result.Add("key1", RawToValue(params.someVar));

It then crashes at:

****************** ASSERT FAILED: Assertion failed in C:\.....\uppsrc_2016\Core\Value.cpp, line 25
ptr()->GetType() >= 255 || !svo[ptr()->GetType()]
which happens to occur in the Value Destructor, ::RefRelease() to be exact.
Why is this happening??

Oh and another thing I already fixed, but it also bothers me:

ValueMap map = rpc["params"]; // rpc = RpcData type
int x = ValueTo<int>(map["someKey"]);

will crash, because it says invalid value convertion, double --> int
Why the heck is map["someKey"] a double value, when it's int?
I fixed it by using:
(int)ValueTo<double>(map["someKey"]);
Not nice... but works.

Thanks again for your help.

Subject: Re: ASSERT when using ValueMap
Posted by NilaT on Wed, 22 Feb 2017 10:56:16 GMT
View Forum Message <> Reply to Message

I got a little bit further when changing my code to:
ValueMap result;
Value v = RawToValue(params.someVar);
result.Add("key1", v);

Then it crashes at the very last line, when I try to Add the whole ValueMap to RpcData with a self written Add method, which adds content not to ValueArray, but to ValueMap from RpcData... like this:
template <class T>

```
void	Add(String key, const T& value)  { Value v; ValuePut(v, value); out_map.Set(key, value); }
```

But it crashes at the same position.
If it helps, I output the type and bool from the condition:
ASSERT(ptr()->GetType() >= 255 || !svo[ptr()->GetType()]); // Check that svo type is not registered as Ref
type = 5 and the bool is false.

Any hints?
Thanks

Subject: Re: ASSERT when using ValueMap
Posted by NilaT on Mon, 27 Feb 2017 15:17:26 GMT
View Forum Message <> Reply to Message

I'm a bit upset that nobody can help me. Posted this almost a week ago and not a single answer from any dev...

I managed to work this issue around by commenting the ASSERT out, but working with RpcData is still a mess to me.
Sometimes rpc["bla"] gets double, when it should be int. Then it gets ValueErrorCls when it's int. Then it gets double when it should be ValueArray... No clue how this piece of **** works.

And the latest error is right here:

```
inline _Uint4_t _Fetch_add_seq_cst_4(volatile _Uint4_t *_Tgt, _Uint4_t _Value)
 { /* add _Value to *_Tgt atomically with
   sequentially consistent memory order */

 return (_INTRIN_SEQ_CST(_InterlockedExchangeAdd)((volatile long *)_Tgt, _Value));
 }
```

WTF is this?!
Really guys, this is messed up...
I simply try to read something out of a JSON and write it back and this is how far I get... absolutely frustrating.
No clue how to solve that and nobody's there who helps...

Subject: Re: ASSERT when using ValueMap
Posted by cbpporter on Tue, 28 Feb 2017 07:46:41 GMT
View Forum Message <> Reply to Message

Hi NilaT,

Unfortunately this is the reality of OSS: sometimes your issue gets addressed in 5 minutes,

sometimes you wait half a year, sometimes it never gets fixed. One week wait time is often barely measurable.

Adding compilable test-cases which show the problem in isolation always help to get you problem in look at sooner rather than later camp.

But back to you problem: ValueMap.Add and RawToValue should be working fine in general since this is something you would instantly notice that is broken. So I would be inclined to say that the problem is what you are passing into RawToValue.

But to be sure, I did a quick test, with someVar being both int and double:

```
ValueMap result;
Value v = RawToValue(someVar);
result.Add("key1", v);
```

So it looks like indeed what you are passing into RawToValue causes the crash. Value implies some sort of copy and that might be the problem. Probably, if you use v in any way for reading it will crash.

To tell you more, I need to see the type of params.someVar, including the prototype of default, copy and move constructors and if there is any assign or move assign constructor. Is the class polymorphic?

Subject: Re: ASSERT when using ValueMap
Posted by cbpporter on Tue, 28 Feb 2017 08:02:17 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Tue, 28 February 2017 09:46Hi NilaT,

Unfortunately this is the reality of OSS: sometimes your issue gets addressed in 5 minutes, sometimes you wait half a year, sometimes it never gets fixed. One week wait time is often barely measurable.

...

cbpporter
Fantastic strike of karma!

No 20 minutes after I wrote this, I got:

Quote:

Assertion failed in C:\SVN\upp\uppsrc\Core\Value.cpp, line 25
ptr()->GetType() >= 255 || !svo[ptr()->GetType()]

in piece of code that ran perfectly for the last few months.

I'm looking into it...

Nevermind. It was the code I used to test your issue, not mine. I inserted it into my real life product code because I had TheIDE open .

So indeed, Value destructor crashes under the scenario you described.

Thank you for reporting it and sorry I didn't catch it the first time I tried to reproduce.

As said, I will look into it.

Subject: Re: ASSERT when using ValueMap
Posted by NilaT on Fri, 03 Mar 2017 08:29:28 GMT
View Forum Message <> Reply to Message

Hi and sorry for my late answer.

Just wanted to say that you can add it to the "bug list", if it really is one.
But a quick fix is not necessary as I changed my JSON/RPC class from Upp to Poco, which works fine.

Thanks anyway for your reply.
Greets

Subject: Re: ASSERT when using ValueMap
Posted by mirek on Mon, 06 Mar 2017 23:43:48 GMT
View Forum Message <> Reply to Message

NilaT wrote on Tue, 21 February 2017 16:23Hello everyone, I've got a nice little "soft crash" again, may you can help me?
It happens when I try to "Add" something to a Value Map.
So my code basically looks like this:

Params params = HandleParams(param1, param2); // Params is a struct which contains some Strings, some ints, and a Time Variable
ValueMap result;

result.Add("key1", RawToValue(params.someVar));

There must be something more to it. I have checked with

ValueMap result;
result.Add("key1", RawToValue(params.someVar));

and that works just fine.

Quote:
Oh and another thing I already fixed, but it also bothers me:

ValueMap map = rpc["params"]; // rpc = RpcData type
int x = ValueTo<int>(map["someKey"]);

will crash, because it says invalid value convertion, double --> int
Why the heck is map["someKey"] a double value, when it's int?

How do you know it is 'int' when reading JSON?

int/double/int64 (and sometimes bool) are used interchangeably. When reading JSON, double is used as "safe option" (because when it is number, it can always be double).

The safe and natural way how to write that is

int x = map["someKey"];

will convert the Value to 'int' when possible.

Mirek

Subject: Re: ASSERT when using ValueMap
Posted by mirek on Mon, 06 Mar 2017 23:58:30 GMT
View Forum Message <> Reply to Message

[quote title=NilaT wrote on Tue, 21 February 2017 16:23]Hello everyone, I've got a nice little "soft crash" again, may you can help me?
It happens when I try to "Add" something to a Value Map.
So my code basically looks like this:

Params params = HandleParams(param1, param2); // Params is a struct which contains some Strings, some ints, and a Time Variable
ValueMap result;
result.Add("key1", RawToValue(params.someVar));

[/code]

OK, found it. Problem like was that params.someVar is likely of some type that is supported as "basic Value" (int, double, int64, String etc...).

E.g. this crashes:

int x;
ValueMap result;
result.Add("key1", RawToValue(x));

That is why nobody noticed this yet, as there is not reason to use RawToValue.

result.Add("key1", x);

works just fine. Anyway, fixed....

Mirek

P.S.: Sorry for the delay, got a bit distracted after last release.

Subject: Re: ASSERT when using ValueMap
Posted by cbpporter on Tue, 07 Mar 2017 09:53:31 GMT
View Forum Message <> Reply to Message

Good you stepped in!

Value is far too complicated to figure out what is inside if you never bothered before.

And this is another example of the U++ "principle" of taking the value of something at an absolute value without minding the consequences.

In U++ I find it that it often holds true "there are some very good arguments for it, so we'll do it". It is almost never "there are some very good arguments for it, but there are also so dire consequences, so...".

Example: "we will put inline functions everywhere on a single line because it saves space and/or

is easy to read for simple functions".

Consequence: the entire U++ codebase has little island of where debugging is very hard.

Nobody puts functions on a single line for this very reason.

Long story short, Value/ValueMap is not debug-able. Period. Whatever arguments one can bring for the single line functions, I really don't think they hold up: what you gain outweigh what you loose.

Subject: Re: ASSERT when using ValueMap
Posted by mirek on Tue, 07 Mar 2017 10:29:32 GMT
View Forum Message <> Reply to Message

It is not inline functions, it is simply the fact that Value code is immensely complicated. Really, absolutely totally fucked hard.

Now the reason why it is so hard is not that I wanted to make everybody looking into it feel like idiot. It is because it is doing some really deep shit optimizations. Value is central to U++ and its perfomance and memory requirements affect everything.

For the record, my yesterday's fix was wrong and I have just spent 4 hours fixing it right (and no, single inline functions were not the problem). And this is just for fixing something that probably was not really in need of fixing - it was borderline to invalid use.

I agree that perhaps more comments would help there. Will try...

Subject: Re: ASSERT when using ValueMap
Posted by mirek on Tue, 07 Mar 2017 10:42:36 GMT
View Forum Message <> Reply to Message

Anyway, this is a new autotest:

```
template <class T>
void Test(T c)
{
 T h = c;

 {
  Value v = h;
  ASSERT(~v == AsString(c));
  ASSERT(v.Is<T>());
  ASSERT(v.To<T>() == h);
```

```
  T h1 = v;
  ASSERT(h == h1);
 }
 {
  Value v = RawToValue(h);
  ASSERT(v.Is<T>());
  ASSERT(v.To<T>() == h);
  T h1 = v.To<T>();
  ASSERT(h == h1);
 }
 { // this is not supported by Value definition (RichToValue should only be used in client type)
  Value v = RichToValue(h);
  ASSERT(~v == AsString(c));
  ASSERT(v.Is<T>());
  T h1 = v.To<T>();
  ASSERT(h == h1);
 }
}

CONSOLE_APP_MAIN
{
 StdLogSetup(LOG_COUT|LOG_FILE);

 {
  struct Foo { int x; } x;
  x.x = 123;
  Value v = RawToValue(x);
  ASSERT(v.Is<Foo>());
  ASSERT(v.To<Foo>().x == 123);
 }

 Test<int>(1234);
 Test<String>("1234");
 Test<WString>("1234");

 LOG("=== Everything OK");
}
```

Subject: Re: ASSERT when using ValueMap
Posted by NilaT on Tue, 07 Mar 2017 10:54:09 GMT
View Forum Message <> Reply to Message

cbpporterLong story short, Value/ValueMap is not debug-able. Period.
mirek it is simply the fact that Value code is immensely complicated. Really, absolutely totally fucked hard.

This may be the most accurate description about U++ ever made!
U++ - it's free, it's good, but it's not documented and it's NOT DEBUG-ABLE REALLY FUCKED HARD CODE WITH SOME REALLY DEEP SHIT OPTIMIZATIONS - download now

Haha laughing so hard right now.
Ok, anyway... Thanks for fixing it, I don't know what RawToValue exactly does, so I use it everywhere. As I understand it, Value is some kind of container which can hold almost anything. But to do so, you have to "convert" it, and that's what RawToValue does... At least that's my belief.

One thing I'm still curious about... What the heck is this assert:
ptr()->GetType() >= 255 || !svo[ptr()->GetType()]
and what does it have in common with
inline _Uint4_t _Fetch_add_seq_cst_4(volatile _Uint4_t *_Tgt, _Uint4_t _Value)
 { /* add _Value to *_Tgt atomically with
   sequentially consistent memory order */

 return (_INTRIN_SEQ_CST(_InterlockedExchangeAdd)((volatile long *)_Tgt, _Value));
 }

Thanks

Subject: Re: ASSERT when using ValueMap
Posted by mirek on Tue, 07 Mar 2017 11:10:20 GMT
View Forum Message <> Reply to Message

NilaT wrote on Tue, 07 March 2017 11:54cbpporterLong story short, Value/ValueMap is not debug-able. Period.
mirek it is simply the fact that Value code is immensely complicated. Really, absolutely totally fucked hard.

This may be the most accurate description about U++ ever made!

Well, I believe not the whole U++. Just this piece of code.

Quote:
but it's not documented.

To be fair, there is shitload of related documentation and examples:

http://www.ultimatepp.org/src$Core$Value$en-us.html

even reading the first couple of senteces would save you a lot of troubles.

http://www.ultimatepp.org/srcdoc$Core$Tutorial$en-us.html

chapter 4 explains everything related to Value.

http://www.ultimatepp.org/reference$JSON$en-us.html

here is how to work with JSON...

I would sort of say that there is perception that documentation is completely missing which was true 5 years ago. Nowadays, there definitely are areas that need improvement, but it is not THAT bad.

Quote:
Ok, anyway... Thanks for fixing it, I don't know what RawToValue exactly does, so I use it everywhere. As I understand it, Value is some kind of container which can hold almost anything. But to do so, you have to "convert" it, and that's what RawToValue does... At least that's my belief.

Well, when type is aware about Value (or Value about type, which is the case of some basic types like 'int'), you do not need any explicit conversions. You are only supposed to use RawToValue for types when implicit conversion does not work.

Quote:
One thing I'm still curious about... What the heck is this assert:
ptr()->GetType() >= 255 || !svo[ptr()->GetType()]
and what does it have in common with

This assert basically checked that you are not using RawToValue for "SVO optimized types". Which I agree is borderline - it might happen this is needed in some template code, so I have changed the rule (I mean, you can now use RawToValue even for types that are alredy supported on higher level).

Quote:
```
inline _Uint4_t _Fetch_add_seq_cst_4(volatile _Uint4_t *_Tgt, _Uint4_t _Value)
 { /* add _Value to *_Tgt atomically with
   sequentially consistent memory order */

 return (_INTRIN_SEQ_CST(_InterlockedExchangeAdd)((volatile long *)_Tgt, _Value));
 }
```

Thanks

Now hard to say about this, but this is probably related to reference counting of one form of internal Value storage. Very likely things got messed up and Value destructor thought it has to release reference count while internal storage was NOT reference counted.

Subject: Re: ASSERT when using ValueMap
Posted by cbpporter on Tue, 07 Mar 2017 12:42:58 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 07 March 2017 12:29It is not inline functions, it is simply the fact that Value code is immensely complicated. Really, absolutely totally fucked hard.

Now the reason why it is so hard is not that I wanted to make everybody looking into it feel like idiot. It is because it is doing some really deep shit optimizations. Value is central to U++ and its perfomance and memory requirements affect everything.

For the record, my yesterday's fix was wrong and I have just spent 4 hours fixing it right (and no, single inline functions were not the problem). And this is just for fixing something that probably was not really in need of fixing - it was borderline to invalid use.

I agree that perhaps more comments would help there. Will try...
Tell me about it! I gave up after 30 minutes of navigating that code when trying to fix the issue.

BTW, I don't know that I ever saw you swear on the forum.

Impostor alert?

Subject: Re: ASSERT when using ValueMap
Posted by mirek on Tue, 07 Mar 2017 12:47:53 GMT
View Forum Message <> Reply to Message

4 hours of fixing 'trivial' issue can do that to you

Subject: Re: ASSERT when using ValueMap
Posted by NilaT on Tue, 07 Mar 2017 13:54:49 GMT
View Forum Message <> Reply to Message

Sorry for being too dumb to fix a 'trivial' error that's easy to understand if you have a master at quantum physics
To be honest, I still have no clue what I did wrong but nevermind, as long as it works now (with poco) everything's fine.

Subject: Re: ASSERT when using ValueMap
Posted by mirek on Tue, 07 Mar 2017 14:00:10 GMT
View Forum Message <> Reply to Message

NilaT wrote on Tue, 07 March 2017 14:54Sorry for being too dumb to fix a 'trivial' error.

Ah, you misunderstood and that makes me sound like I am ranting...

Error was 'borderline' - by that I mean that it was very unusual usage (there reading docs would help). Anyway, I decided it should work even if used that atypical way, so had to fix that. It was not anything you were supposed to fix, that was my task.

It looked 'trivial' to fix at first, but things were more complicated than I anticipated.

What you did 'wrong' is that you have used RawToValue and ValueTo without reason and perhaps without complete understanding what they mean...

```
Params params = HandleParams(param1, param2); // Params is a struct which contains some Strings, some ints, and a Time Variable
ValueMap result;
result.Add("key1", params.someVar);
```

```
ValueMap map = rpc["params"]; // rpc = RpcData type
int x = map["someKey"];
```

would have worked (I supposed someVar is String or int or something like that).

Mirek

Subject: Re: ASSERT when using ValueMap
Posted by NilaT on Tue, 07 Mar 2017 14:49:21 GMT
View Forum Message <> Reply to Message

mirekwould have worked
Alright, thanks
Sorry for not testing it though.
