Subject: Jsonize int64 surprise
Posted by Mindtraveller on Thu, 24 Jan 2013 18:51:36 GMT
View Forum Message <> Reply to Message

It looks like jsonizing big int64 numbers leads to "string" representation instead of direct one.
I consider this bad kind of surprise as one's completely unable to send ANY big (int64) numbers to browser. I.e. you need to send some JSON containing javascript timestamp (which is really big int64 number) with int64 member. But you'll be surprised that this javascript code wont't work:
var d = new Date(receivedJson.t);
because t is not 1274574567221 but is a string '1274574567221'.

Looking into U++ core code reveals following:

```
//Core/JSON.cpp @ 238
if(var >= INT_MIN && var <= INT_MAX)
   io.Set(var);
  else
   io.Set(AsString(var));
```

I propose eliminating such a surprise party leaving int64 numbers as it is. Besides all known to me modern javascript implementations DO SUPPORT int64 integer numbers by default.

Subject: Re: Jsonize int64 surprise
Posted by mirek on Thu, 24 Jan 2013 19:02:52 GMT
View Forum Message <> Reply to Message

I do not know. Quick internet search reveals that JSON format in principle cannot support int64. E.g.

http://json.codeplex.com/workitem/20832

Anyway, what we perhaps COULD do it to increase limits to accomodate 'precise integer' range of double (that is, if I remember well, 56-bits). That would be JSON compliant.

Mirek

Subject: Re: Jsonize int64 surprise
Posted by Mindtraveller on Thu, 24 Jan 2013 19:21:49 GMT
View Forum Message <> Reply to Message

I may be wrong. But my point is that JSON is used mostly in Web development. In our case this means: Quote:server[U++] <-> client[javascript]
And javascript natively supports integers up to 64-bit length (actually 53-bit, see below). So should we in U++.
This article

http://cdivilly.wordpress.com/2012/04/11/json-javascript-lar ge-64-bit-integers/
corrects that Javascript actually support integers up to 2^53. So if we really want to be close to standards here, we should check not INT_MAX, but 2^53 for stringification.

But I'd suggest eliminating ANY surprises at all. We have number - we represent as number. But one may disagree with it.

---

Subject: Re: Jsonize int64 surprise
Posted by mirek on Thu, 24 Jan 2013 19:30:22 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Thu, 24 January 2013 14:21I may be wrong. But my point is that JSON is used mostly in Web development. In our case this means: Quote:server[U++] <-> client[javascript] And javascript natively supports integers up to 64-bit length (actually 53-bit, see below). So should we in U++.
This article
http://cdivilly.wordpress.com/2012/04/11/json-javascript-lar ge-64-bit-integers/
corrects that Javascript actually support integers up to 2^53. So if we really want to be close to standards here, we should check not INT_MAX, but 2^53 for stringification.

Well, but that actually is exactly what I propose (I just got 56 wrong, it is indeed 53).

Quote:
But I'd suggest eliminating ANY surprises at all. We have number - we represent as number. But one may disagree with it.

Well, there seems to be no 100% correct solution, but I think that lost precision would not be a good thing as well...

Mirek

---

Subject: Re: Jsonize int64 surprise
Posted by Mindtraveller on Thu, 24 Jan 2013 19:43:59 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 24 January 2013 22:30that actually is exactly what I propose (I just got 56 wrong, it is indeed 53).
Excellent. So here's change I propose:
//JSON.cpp @ 238
```
 {
  static const int64 JSON_INT_MIN = -9007199254740992LL;
  static const int64 JSON_INT_MAX =  9007199254740991LL;

  if(var >= JSON_INT_MIN && var <= JSON_INT_MAX)
```

```
    io.Set(var);
  else
    io.Set(AsString(var));
 }
```

Subject: Re: Jsonize int64 surprise
Posted by mirek on Thu, 24 Jan 2013 20:06:34 GMT
View Forum Message <> Reply to Message

Commited.

Mirek

Subject: Re: Jsonize int64 surprise
Posted by sergeynikitin on Tue, 29 Jan 2013 07:06:02 GMT
View Forum Message <> Reply to Message

By the way same situation with int64 in XMLize class!

Subject: Re: Jsonize int64 surprise
Posted by mirek on Tue, 29 Jan 2013 07:29:27 GMT
View Forum Message <> Reply to Message

sergeynikitin wrote on Tue, 29 January 2013 02:06By the way same situation with int64 in XMLize class!

That is strange, at the first glance it looks ok (mostly because in XML, you need to store it as string no matter what...)

Mirek

Subject: Re: Jsonize int64 surprise
Posted by sergeynikitin on Tue, 29 Jan 2013 09:37:23 GMT
View Forum Message <> Reply to Message

Check pls Sql for int64 columns also.
(I've spent many many hours to discover problem, that int64 is String Value)

Subject: Re: Jsonize int64 surprise

Posted by mirek on Tue, 29 Jan 2013 09:44:52 GMT
View Forum Message <> Reply to Message

sergeynikitin wrote on Tue, 29 January 2013 04:37Check pls Sql for int64 columns also.
(I've spent many many hours to discover problem, that int64 is String Value)

Sql depends on database used. Some might have issue...

Mirek

Subject: Re: Jsonize int64 surprise
Posted by sergeynikitin on Tue, 29 Jan 2013 20:59:11 GMT
View Forum Message <> Reply to Message

I've test with MySQL.

Subject: Re: Jsonize int64 surprise
Posted by mirek on Wed, 30 Jan 2013 07:15:43 GMT
View Forum Message <> Reply to Message

Confirmed and filed into RM.

Mirek

Subject: Re: Jsonize int64 surprise
Posted by sergeynikitin on Wed, 30 Jan 2013 08:01:47 GMT
View Forum Message <> Reply to Message

And may you patch Sqlite3 schema plugin to do not convert int64 to int Type?

Sqlite normally eat int64 declaration, but if I write universal SqlExp expression - Sqlite3 schemaparser convert int64 to int. It's sad!

uppsrc/plugin/sqlite3/Sqlite3Schema.h:
Now:#define INT64(x) COLUMN("integer", int64, x, 0, 0)
#define INT64_ARRAY(x, items) COLUMN_ARRAY("integer", int64, x, 0, 0, items)
#define INT64_(x) COLUMN_("integer", int64, x, 0, 0)
#define INT64_ARRAY_(x, items) COLUMN_ARRAY_("integer", int64, x, 0, 0, items)

Need and tested: #define INT64(x) COLUMN("bigint", int64, x, 0, 0)
#define INT64_ARRAY(x, items) COLUMN_ARRAY("bigint", int64, x, 0, 0, items)
#define INT64_(x) COLUMN_("bigint", int64, x, 0, 0)
#define INT64_ARRAY_(x, items) COLUMN_ARRAY_("bigint", int64, x, 0, 0, items)