

---

Subject: Re: ValueArray behaviour / inconsistantcy / BUG?

Posted by [kohait00](#) on Wed, 20 Oct 2010 08:45:30 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

i've been debugging a bit, and am pretty sure thats a bug in ValueArray:

having refactored my Xmlize procedure to

```
#include <Core/Core.h>
```

```
using namespace Upp;
```

```
void ValueArrayXmlize(XmlIO xml, Value& v)
```

```
{
  if(xml.IsStoring())
  {
    const ValueArray& va = v;
    XmlizeStore(xml, va.Get());
  }
  if(xml.IsLoading())
  {
    Vector<Value> vv;
    ::Xmlize(xml, vv);
    v = ValueArray(vv);
  }
}
INITBLOCK { RegisterValueXmlize(GetValueTypeNo<ValueArray>(), &ValueArrayXmlize,
"ValueArray"); }
```

```
CONSOLE_APP_MAIN
```

```
{
  Vector<Value> vv;
  vv << 123;
  vv << "Hallo";
```

```
  ValueArray va(vv); //picks vv, ValueArray::Data contains the Vector, ValueArray ref count
  references the VA::Data
```

```
  Value v = va; //the Value now additionally references the ValueArray::Data
```

```
  RLOG(v);
```

```
  ValueArray va_ = v; //now another ValueArray references the same ValueArray::Data
  const Vector<Value>& vv_ = va_.Get();
  DUMPC(vv_);
```

```
  StoreAsXMLFile(v); //the pure Value is xmlized with the ValueArray::Data as Void derived
  content, not as RichValue
```

```
StoreToFile(v); //the pure Value is serialized with the ValueArray::Data as Void derived content,
not as RichValue
```

```
Value v2;
LoadFromXMLFile(v2); //the Value is dexmlized, with help of ValueArray to only create the ::Void
derived Container
RLOG(v2);
```

```
//some checks
ASSERT(IsValueArray(v2)); //is OK
const ValueArray& va2 = v2;
const Vector<Value>& vv2 = va2.Get();
ASSERT(va2.GetCount() == va.GetCount());
for(int i = 0; i < vv2.GetCount(); i++)
    ASSERT(vv2[i] == va[i]);
```

```
Value v3;
LoadFromFile(v3); //is deserilized as RichValue, with ValueArray as content, wich has
ValueArray::Data as its content
RLOG(v3); //works fine anyway
```

```
ASSERT(IsValueArray(v3)); //still no problem, though not consistent.
const ValueArray& va3 = v3; //CRASH, here comes the inconsistency
const Vector<Value>& vv3 = va3.Get();
ASSERT(va3.GetCount() == va.GetCount());
for(int i = 0; i < vv3.GetCount(); i++)
    ASSERT(vv3[i] == va[i]);
}
```

it works as expected, as far as i understand.

now the question is, how is ValueArray supposed to be used:

is it supposed to be used as

- 1) Content of a RichValueRep? (Value.cpp:142: RichValue<ValueArray>::Register())
- 2) as a selfsustained Value interface/data container, just same as RawValue / RichValue?

as far as i got it to understand, ValueArray is kind of both. an extended Value implementation and meant to be in container data of a normal value, but offers the ValueArray::Data : Value::Void implementation, which is sort of Value domain.

now converting a ValueArray to a Value takes over the ValueArray::Data, serilizing it properly. deserializing it creates a RichValueRep<ValueArray>, which is inconsistent.

>>> so how is ValueArray supposed to be handled? how to deal with the inconsistency? i'd suggest to make ValueArray / ValueMap a true Value derive...not a RichValue related one.. this comes closer to the idea of the ::Void derived refcounted ValueArray::Data

here, an updated testcase

## File Attachments

---

1) [ValueArrayTest.rar](#), downloaded 250 times

---