

## TYPO3.Flow - Bug # 47975

<b>Status:</b>	Resolved	<b>Priority:</b>	Must have
<b>Author:</b>	Benno Weinzierl	<b>Category:</b>	Object
<b>Created:</b>	2013-05-06	<b>Assigned To:</b>	Robert Lemke
<b>Updated:</b>	2014-07-11	<b>Due date:</b>	
<b>PHP Version:</b>	5.4		
<b>Has patch:</b>	No		
<b>Complexity:</b>			
<b>Affected Flow version:</b>	Git master		
<b>Subject:</b>	initializeObject in a Entity is called at a time where no properties are loaded		
<b>Description</b>	<p>I have an Entity which is loaded during validation of a Action Argument. (The Action Argument is another Entity with relations to the Entity in question)</p> <p>When initializeObject() is called all properties are missing and have not been loaded from persistence.</p> <p>Here is the Backtrace: <a href="https://gist.github.com/anonymous/5524256">https://gist.github.com/anonymous/5524256</a></p>		
<b>Related issues:</b>	duplicates TYPO3.Flow - Bug # 43659: Proxy class building calls __construct a... <b>Resolved</b> <b>2012-12-06</b>		

### History

#### #1 - 2013-05-06 17:29 - Alexander Berl

The real problem is, that the Doctrine Proxy \_\_load() calls \_\_wakeup() before the properties are actually loaded.

Here's a pseudo call stack for the class hierarchy of the proxies to original class, when the doctrine proxy is the entry point:

Class hierarchy: Doctrine Proxy (DP) > Flow Proxy (FP) > Original Class (OC)

Call stack1: Doctrine Proxy gets loaded due to property access or via the GenericObjectValidator

```
DP->__load()
  DP->__wakeup()
    DP->__load()
      FP->__wakeup()
        OC->__wakeup()
          DP->initializeObject(2)
            DP->__load()
              OC->initializeObject()
DP->_entityPersister->load($this->_identifier, $this)
```

Call stack2: Doctrine Proxy gets unserialized (e.g. unserialize('O:...') hacks used both in Flow and Doctrine)

```
DP->__wakeup()
  DP->__load() [I think this is not really executed, since $this->_entityPersister would be empty in an empty unserialization]
    FP->__wakeup()
      OC->__wakeup()
        DP->initializeObject(2)
          DP->__load()
            OC->initializeObject()
```

Indentation refers to the call nesting level. As you can see, in both cases the initializeObject method gets called before the object properties are loaded. However, I'm not sure how to best deal with the premature \_\_wakeup() in \_\_load() and preventing empty unserialization wakeups from triggering initializeObject.

A possible check could be that if \_\_isInitialized\_\_ is set and true and \_\_entityPersister is set, or if \_\_isInitialized\_\_ is set and false then skip the initializeObject invocation in the FlowProxy. You'd completely lose the initializeObject call in those cases though and at least in the first this is not desired I guess.

## #2 - 2013-05-09 18:55 - rottenrice no-lastname-given

I have the same problem but my call stack differs from your call stack.

My two Entities:

```
/** @Flow\Entity */
class Foo {
    public function __construct() {
        echo '__construct()';
    }
    public function initializeObject($arg) {
        echo 'initializeObject('.$arg.')';
    }
    public function injectSettings() {
        echo 'injectSettings()';
    }
}
```

```
/** @Flow\Entity */
class Bar extends Foo {
    public function __construct() {
        echo '__construct()';
        parent::__construct();
    }
}
```

Example code:

```
$bar = new Bar();
```

Output:

```
__construct()
__construct()
initializeObject(1)
injectSettings()
initializeObject(1)
```

Reason:

\_\_construct() of Flow-Proxy (Foo)

[...]

```
if ('Foo' === get_class($this)) {  
    $this->Flow_Proxy_injectProperties();  
}  
$this->initializeObject(1);
```

The injectProperties() is not executed but the initializeObject() is executed :/

### #3 - 2013-05-14 17:47 - Adrian Förder

I also had that; and I realized it's because of this gem here:

```
\Doctrine\ORM\Mapping\ClassMetadataInfo::newInstance  
  
1 public function newInstance()  
2 {  
3     if ($this->_prototype === null) {  
4         $this->_prototype = unserialize(sprintf("O:%d: \"%s\":0:{}", strlen($this->name), $this->name));  
5     }  
6  
7     return clone $this->_prototype;  
8 }
```

This is a dummy way to get an object without the constructor being called; but unfortunately this calls \_\_wakeup and hence initializeObject(), of course without any properties then.

At the end, I had initializeObject() being called twice, once in that erroneous case and once when it's actually intended.

### #4 - 2013-05-21 11:31 - Robert Lemke

- Status changed from New to Accepted

Needs to be verified and solved for 2.0

### #5 - 2013-08-14 15:35 - Karsten Dambekalns

- Target version changed from 2.0 to 2.0.1

### #6 - 2013-10-11 10:10 - Robert Lemke

- Category changed from Persistence to Object

- Assigned To set to Robert Lemke

- PHP Version set to 5.4

**#7 - 2013-10-11 10:12 - Robert Lemke**

I think that this is a duplicate of #43659

**#8 - 2013-10-11 10:13 - Robert Lemke**

- *Status changed from Accepted to Closed*

Let's see if the fix for #43659 helps

**#9 - 2013-10-11 11:03 - Andreas Förthner**

- *Status changed from Closed to New*

The problem still exists, opening the issue again...

**#10 - 2014-07-11 20:19 - Robert Lemke**

- *Status changed from New to Resolved*

Applied in changeset commit:6d2ea6b3fc5db90f3c7e6ccd0dea5304ec32ef5e.