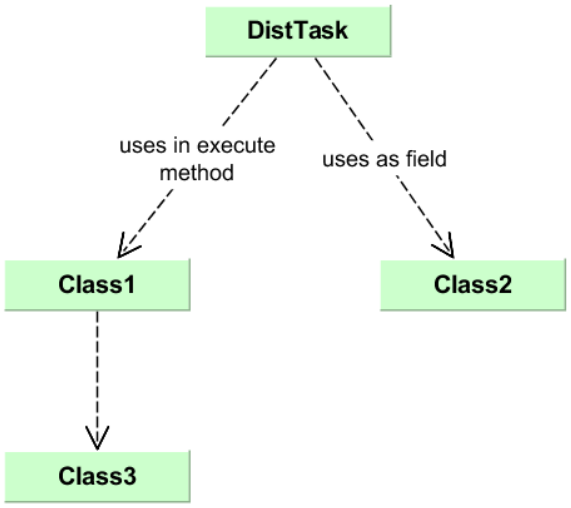
Changing implementation of dist task classes without restarting space

1. Issue description



We are running many dist tasks on remote spaces. Lifecycles of stateless PUs and spaces PUs from deployment/release perspective are different. We release/deploy stateless PUs more frequently.
Whenever we change implementation of dist task or any class used by dist task we can face following issues:

- when state of the class (dist task or dependent class) is changed we get serialization exception because space PU GSC hold definition of old class
- when behavior of the class (dist task or dependent class) is change (without changing state) we execute old business logic and we are not aware what is very dangerous

1.1. How to reproduce

1.1.1. State changed

1. deploy space PU
2. run dist task with serialVersionUID=1 from stateless pu or any java client
3. change serialVersionUID=2 of dist task
4. run dist task, then you get InvalidClassException

```
2016-09-26 13:04:54,847 GSC SEVERE [com.gigaspaces.lrmi] - LRMI Transport Protocol caught server
exception caused by [/14.69.22.146:65122] client.; Caused by:
com.gigaspaces.lrmi.nio.UnMarshallingException: Failed to unmarsh :[RequestPacket: interface
com.gigaspaces.internal.remoting.RemoteOperationsExecutor.executeOperationAsync(com.gigaspaces.intern
null), isOneWay = false, isCallBack = false, Priority = CUSTOM]
        at com.gigaspaces.lrmi.nio.Reader.unmarshall(Reader.java:641)
        at com.gigaspaces.lrmi.nio.Reader.unmarshallRequest(Reader.java:525)
        at com.gigaspaces.lrmi.nio.ChannelEntry.unmarshall(ChannelEntry.java:158)
        at com.gigaspaces.lrmi.nio.Pivot$ChannelEntryTask.run(Pivot.java:182)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
        at java.lang.Thread.run(Thread.java:744)
Caused by: java.io.InvalidClassException: classloading.DistTaskNotLoadingAllClasses; local class
incompatible: stream classdesc serialVersionUID = 2, local class serialVersionUID = 1
...
```

1.1.2. Behavior changed

1. deploy space PU
2. run dist task with with dependent class Class1, serialVersionUID=1 from stateless pu or any java client
3. change implementation of Class1, (serialVersionUID could be change as well but it does not matter since class is not part of dist task state, only used in execute method)
4. run dist task, old version of Class1 is used

2. How we workaround this issue currently

Whenever we release new version of stateless PU we transform sources of all classes used in dist task to have unique names, e.g.

```
com.ubs.opsit.__version__.DistTask        ->  com.ubs.opsit.v_18_561.DistTask
com.ubs.opsit.__version__.DependentClass  ->  com.ubs.opsit.v_18_561.DependentClass
```

This transformation is done before compilation and testing phase.

Possible issue with this solution is that old classes are loaded and consume memory of spaces GSC. Additionally class package in source code is a bit weird, also we do this transformation only for classes used in dist task so there is some mess.

3. Solutions/Ideas worth consideration

Whenever any class in dist task classes graph is changed we update serialVersionUID of dist task. GS reloads all classes (or discard old classes) used by dist task whenever serialVersionUID is changed compared to currently loaded class.
Of course, this solution is error prone because developer can change some dependent class (e.g. Class3 on diagram) and forget to change serialVersionUID on dist task but we can use simple source transformation and update serialVersionUID each time with release (serialVersionUID value correlates to release version).