Foswiki > FLAO Web > SoftwareDevelopment > TextModeSerialization (09 Apr 2015, AlfioPuglisi)

Text mode serialization

When the LBTO team decided to move the TCS to 64 bit machines, we discovered the incompatibility of message serialization algorithms in the boost library (used for the communication with the arbitrators, both from AOS and internally to FLAO). We thus decided to adopt "text mode" for the serialization of messages which would allow the intercommunication of AOS (64 bit) with 32 bit FLAO Supervisor. When we tested the text serialization mode we also discovered that the standard boost library does not support serialization of *NaN* or *Inf* values, so we added some code to manage the case.

Here follows a resume of the changes to be made to FLAO software in order to use text mode serialization.

What was modified

The modification of source code covers two aspects:

1. The substitution of two functions: `binary_iarchive` and `binary_oarchive` with the corresponding `text_iarchive` and `text_oarchive`. This was done by using an `#ifdef` construct.
2. Using a proper wrapper for float/double values in order to serialize properly *inf* and *NaN* values which are used in the code.

Then we modified some makefiles in order to allow the selection of the wanted serialization mode.

Note 1: The modifications in the FLAO code was originally made with conditional compilation because we did not know whether the modified code would work before testing. Now the code seems stable and we do not see reasons to go back to binary serialization in the future, so furter code can be modified more easily doing the simple substitutions quoted above.

Note 2: The reference SVN branch for the code is:

svn+ssh://adopt.arcetri.astro.it/4lbt/svn/AOSupervisor/branches/adsecSX2

Most of the files can be retrieved from there after an inspection to see if there are modifications other than those described below.

Affected files

For each we report the output of diff command between two revision (before and after the modification). Please note that some details may differ if your code is in a different revision, but the diff should show the required modification pretty well, anyway.

./makefile.gen

```
-CFLAGS = -W -Wall -Wreturn-type -Wunused -Wmissing-prototypes -D_GNU_SOURCE
-CPPFLAGS = -Wall -W -Wreturn-type -Wunused -D_GNU_SOURCE
+# Get wanted serialization mode from getserializationflags.py
+BOOST_SERIALIZATION = $(shell python
$(ADOPT_SOURCE)/getserializationflags.py)
+
+
+CFLAGS = -W -Wall -Wreturn-type -Wunused -Wmissing-prototypes -D_GNU_SOURCE
$(BOOST_SERIALIZATION)
+CPPFLAGS = -Wall -W -Wreturn-type -Wunused -D_GNU_SOURCE
$(BOOST_SERIALIZATION)
```

./GUI/cc.pro

```
QMAKE_LINK = $(Q)echo "  [LD] $@"; gcc

+QMAKE_CXXFLAGS += $(shell python $(ADOPT_SOURCE)/getserializationflags.py)
+
```

./lib/aoapplib.cpp

```
--- lib/aoapplib.cpp   (revision 3700)
+++ lib/aoapplib.cpp   (revision 3848)
@@ -10,7 +10,11 @@
     LogLevelModifier llm(logger,level);

     std::ostringstream oss;
+#ifdef TEXT_SERIALIZATION
+    boost::archive::text_oarchive oa(oss);
+#else
     boost::archive::binary_oarchive oa(oss);
+#endif
     oa << llm;

     std::string sbuf = oss.str();
@@ -54,7 +58,11 @@
             {
             unsigned int len = HDR_LEN(msgb);
             std::istringstream iss( std::string( (const char *)
(MSG_BODY(msgb)), len));
+#ifdef TEXT_SERIALIZATION
+            boost::archive::text_iarchive ia(iss);
+#else
             boost::archive::binary_iarchive ia(iss);
+#endif
             ia >> ret;
             thRelease(msgb);
             break;
```

./lib/AOApp.cpp

```
--- lib/AOApp.cpp   (revision 3700)
+++ lib/AOApp.cpp   (revision 3848)
@@ -628,7 +628,11 @@
 // See @see{InstallHandlers}.
 //@

+#ifdef TEXT_SERIALIZATION
+#include <boost/archive/text_iarchive.hpp>
+#else
 #include <boost/archive/binary_iarchive.hpp>
```

```
+#endif
 #include <sstream>
 #include <string>
 #include "aoapplib.h"
@@ -642,7 +646,11 @@
     unsigned int len = HDR_LEN(msgb);
     std::string sbuf( (const char *)(MSG_BODY(msgb)), len);
     std::istringstream iss(sbuf);
+#ifdef TEXT_SERIALIZATION
+    boost::archive::text_iarchive ia(iss);
+#else
     boost::archive::binary_iarchive ia(iss);
+#endif
     ia >> llm;
     if ( Logger::exists(llm.Logger()) ) {
         Logger::get(llm.Logger())->setGlobalLevel(llm.Level());
@@ -663,7 +671,11 @@
 // See @see{InstallHandlers}.
 //@

+#ifdef TEXT_SERIALIZATION
+#include <boost/archive/text_oarchive.hpp>
+#else
 #include <boost/archive/binary_oarchive.hpp>
+#endif
 #include <sstream>
 #include <string>
 #include "aoapplib.h"
@@ -675,7 +687,11 @@


     std::ostringstream oss;
+#ifdef TEXT_SERIALIZATION
+    boost::archive::text_oarchive oa(oss);
+#else
     boost::archive::binary_oarchive oa(oss);
+#endif
     oa << mappa;

     std::string outbuf = oss.str();
```

./lib/aoapplib.h

```
--- lib/aoapplib.h   (revision 3700)
+++ lib/aoapplib.h   (revision 3848)
@@ -6,8 +6,13 @@
 #define AOAPPLIB_H_INCLUDED

 #include "AOExcept.h"
+#ifdef TEXT_SERIALIZATION
+#include <boost/archive/text_oarchive.hpp> //serialize
+#include <boost/archive/text_iarchive.hpp> //serialize
+#else
 #include <boost/archive/binary_oarchive.hpp> //serialize
 #include <boost/archive/binary_iarchive.hpp> //serialize
+#endif
```

./lib/arblib/adSecArb/AdSecPTypes.h

```
--- lib/arblib/adSecArb/AdSecPTypes.h   (revision 3700)
+++ lib/arblib/adSecArb/AdSecPTypes.h   (revision 3848)
@@ -3,8 +3,13 @@

 #include <string>

+#ifdef TEXT_SERIALIZATION
+#include <boost/archive/text_iarchive.hpp>
+#include <boost/archive/text_oarchive.hpp>
+#else
 #include <boost/archive/binary_iarchive.hpp>
 #include <boost/archive/binary_oarchive.hpp>
+#endif
```

./lib/arblib/adSecArb/AdSecCommands.h

```
--- lib/arblib/adSecArb/AdSecCommands.h   (revision 3700)
+++ lib/arblib/adSecArb/AdSecCommands.h   (revision 3848)
@@ -5,6 +5,7 @@
```

```
 #include "arblib/adSecArb/AdSecPTypes.h"

 #include "arblib/base/Commands.h"
+#include "AONanWrapper.hpp"

 //using namespace Arcetri::Arbitrator;

@@ -125,8 +126,28 @@
       bool validateImpl() throw(CommandValidationException);

       friend class boost::serialization::access;
+#ifdef TEXT_SERIALIZATION
        template<class Archive>
       void serialize(Archive& ar, const unsigned int /* version */) {
+                       AONanWrapper<float>
__safeSkipPercent(_safeSkipPercent);
+                       AONanWrapper<double> __tipOffload(_tipOffload),
__tiltOffload(_tiltOffload), __focusOffload(_focusOffload);
+
+         // Serialize base class object
+         ar & boost::serialization::base_object<RequestStatus>(*this);
+         ar & _safeSkipFramesCounter;
+         ar & __safeSkipPercent;
+         ar & _isSkippingFrames;
+         ar & _clStatReady;
+         ar & _coilsEnabled;
+         ar & _tssEnabled;
+         ar & __tipOffload;
+         ar & __tiltOffload;
+         ar & __focusOffload;
+      }
+#else
+       template<class Archive>
+      void serialize(Archive& ar, const unsigned int /* version */) {
+
          // Serialize base class object
          ar & boost::serialization::base_object<RequestStatus>(*this);
          ar & _safeSkipFramesCounter;
@@ -139,6 +160,7 @@
          ar & _tiltOffload;
          ar & _focusOffload;
```

```
       }
+#endif

 };
```

./lib/arblib/aoArb/AOPTypes.h

```
--- lib/arblib/aoArb/AOPTypes.h   (revision 3700)
+++ lib/arblib/aoArb/AOPTypes.h   (revision 3848)
@@ -3,11 +3,17 @@

 #include <string>

+#ifdef TEXT_SERIALIZATION
+#include <boost/archive/text_iarchive.hpp>
+#include <boost/archive/text_oarchive.hpp>
+#else
 #include <boost/archive/binary_iarchive.hpp>
 #include <boost/archive/binary_oarchive.hpp>
+#endif

 #include "AOGlobals.h"
 #include "AOArbConst.h"
+#include "AONanWrapper.hpp"

 using namespace std;

@@ -27,10 +33,18 @@
     private:

        friend class boost::serialization::access;
+#ifdef TEXT_SERIALIZATION
       template<class Archive>
       void serialize(Archive& ar, const unsigned int /* version */) {
-         ar & gain;
+      AONanWrapper<double> _gain(gain);
+      ar & _gain;
+      }
+#else
+      template<class Archive>
```

```
+       void serialize(Archive& ar, const unsigned int /* version */) {
+       ar & gain;
        }
+#endif
 } adjustGainParams;

 /*
@@ -43,10 +57,18 @@
     private:

         friend class boost::serialization::access;
+#ifdef TEXT_SERIALIZATION
+       template<class Archive>
+       void serialize(Archive& ar, const unsigned int /* version */) {
+       AONanWrapper<double> _intTime(intTime);
+       ar & _intTime;
+       }
+#else
        template<class Archive>
        void serialize(Archive& ar, const unsigned int /* version */) {
-          ar & intTime;
+       ar & intTime;
        }
+#endif
 } adjustIntTimeParams;


@@ -110,8 +132,36 @@
     private:

         friend class boost::serialization::access;
+
+#ifdef TEXT_SERIALIZATION
        template<class Archive>
        void serialize(Archive& ar, const unsigned int /* version */) {
+       AONanWrapper<double> _soCoord[2] = {soCoord[0], soCoord[1]};
+       AONanWrapper<double> _roCoord[2] = {roCoord[0], roCoord[1]};
+       AONanWrapper<double> _elevation(elevation), _rotAngle(rotAngle),
_gravAngle(gravAngle);
+       AONanWrapper<double> _mag(mag), _color(color), _r0(r0);
+       AONanWrapper<double> _skyBrgt(skyBrgt), _windSpeed(windSpeed),
```

```
_windDir(windDir);
+
+         ar & aoMode;
+         ar & focStation;
+         ar & instr;
+         ar & wfsSpec;
+         ar & _soCoord;
+         ar & _roCoord;
+         ar & _elevation;
+         ar & _rotAngle;
+         ar & _gravAngle;
+         ar & _mag;
+         ar & _color;
+         ar & _r0;
+         ar & _skyBrgt;
+         ar & _windSpeed;
+         ar & _windDir;
+      }
+#else
+      template<class Archive>
+      void serialize(Archive& ar, const unsigned int /* version */) {
+
          ar & aoMode;
          ar & focStation;
          ar & instr;
@@ -128,6 +178,8 @@
          ar & windSpeed;
          ar & windDir;
       }
+#endif
+
 } presetAOParams;


@@ -162,6 +214,36 @@
     private:

        friend class boost::serialization::access;
+#ifdef TEXT_SERIALIZATION
+      template<class Archive>
+      void serialize(Archive& ar, const unsigned int /* version */) {
```

```
+      AONanWrapper<double> _deltaXY[2] = {deltaXY[0], deltaXY[1]};
+      AONanWrapper<double> _freq(freq), _ttMod(ttMod), _gain(gain);
+      AONanWrapper<double> _strehl(strehl), _r0(r0), _starMag(starMag),
_CLfreq(CLfreq);
+         ar & _deltaXY;
+         ar & slNull;
+         ar & nModes;
+         ar & _freq;
+         ar & nBins;
+         ar & _ttMod;
+         ar & f1spec;
+         ar & f2spec;
+         ar & _gain;
+         ar & pupils;
+         ar & base;
+         ar & rec;
+         ar & filtering;
+         ar & decimation;
+         ar & _strehl;
+         ar & _r0;
+         ar & mSNratio;
+         ar & _starMag;
+         ar & CLbase;
+         ar & CLrec;
+         ar & _CLfreq;
+         ar & TVframe;
+      }
+#else
       template<class Archive>
       void serialize(Archive& ar, const unsigned int /* version */) {
          ar & deltaXY;
@@ -187,6 +269,7 @@
          ar & CLfreq;
          ar & TVframe;
       }
+#endif

 } acquireRefAOResult;

@@ -202,12 +285,21 @@
     private:
```

```
      friend class boost::serialization::access;
-
+#ifdef TEXT_SERIALIZATION
+     template<class Archive>
+     void serialize(Archive& ar, const unsigned int /* version */) {
+         AONanWrapper<double> _deltaXY[2] = {deltaXY[0], deltaXY[1]};
+         AONanWrapper<double> _starMag(starMag);
+         ar & _deltaXY;
+         ar & _starMag;
+     }
+#else
      template<class Archive>
      void serialize(Archive& ar, const unsigned int /* version */) {
          ar & deltaXY;
          ar & starMag;
      }
+#endif

 } checkRefAOResult;

@@ -224,10 +316,18 @@

      friend class boost::serialization::access;

+#ifdef TEXT_SERIALIZATION
+     template<class Archive>
+     void serialize(Archive& ar, const unsigned int /* version */) {
+         AONanWrapper<double> _deltaXY[2] = {deltaXY[0], deltaXY[1]};
+         ar & _deltaXY;
+     }
+#else
      template<class Archive>
      void serialize(Archive& ar, const unsigned int /* version */) {
          ar & deltaXY;
      }
+#endif

 } offsetXYParams;

@@ -243,11 +343,18 @@
```

```
     private:

      friend class boost::serialization::access;
-
+#ifdef TEXT_SERIALIZATION
+     template<class Archive>
+     void serialize(Archive& ar, const unsigned int /* version */) {
+         AONanWrapper<double> _deltaZ(deltaZ);
+         ar & _deltaZ;
+     }
+#else
      template<class Archive>
      void serialize(Archive& ar, const unsigned int /* version */) {
          ar & deltaZ;
      }
+#endif

 } offsetZParams;

@@ -268,7 +375,18 @@
     private:

      friend class boost::serialization::access;
-
+#ifdef TEXT_SERIALIZATION
+     template<class Archive>
+     void serialize(Archive& ar, const unsigned int /* version */) {
+         AONanWrapper<double> _freq(freq), _TTmod(TTmod);
+         ar & nModes;
+         ar & _freq;
+         ar & binning;
+         ar & _TTmod;
+         ar & f1spec;
+         ar & f2spec;
+     }
+#else
      template<class Archive>
      void serialize(Archive& ar, const unsigned int /* version */) {
          ar & nModes;
@@ -278,6 +396,7 @@
          ar & f1spec;
```

```
          ar & f2spec;
      }
+#endif

 } modifyAOParams;

@@ -292,7 +411,6 @@
     private:

      friend class boost::serialization::access;
-
      template<class Archive>
      void serialize(Archive& ar, const unsigned int /* version */) {
          ar & deltaM;
```

./lib/arblib/base/Serializator.cpp

```
--- lib/arblib/base/Serializator.cpp   (revision 3700)
+++ lib/arblib/base/Serializator.cpp   (revision 3848)
@@ -4,8 +4,13 @@
 using namespace Arcetri::Arbitrator;

 #include <boost/serialization/base_object.hpp>
+#ifdef TEXT_SERIALIZATION
+#include <boost/archive/text_iarchive.hpp>
+#include <boost/archive/text_oarchive.hpp>
+#else
 #include <boost/archive/binary_iarchive.hpp>
 #include <boost/archive/binary_oarchive.hpp>
+#endif
 #include <boost/serialization/nvp.hpp>

 char* Serializator::serialize(SerializableMessage* msg, int& bufLen) {
@@ -13,12 +18,16 @@
     char* buf = NULL;
     bufLen = 0;

-    // Serialize the message into an outoput stream
+    // Serialize the message into an output stream
     try {
```

```
       _logger->log(Logger::LOG_LEV_DEBUG, "Serializing message:");
       msg->log();
       ostringstream serializedMsg;
+#ifdef TEXT_SERIALIZATION
+      boost::archive::text_oarchive archive(serializedMsg);
+#else
       boost::archive::binary_oarchive archive(serializedMsg);
+#endif
       archive << BOOST_SERIALIZATION_NVP(msg);
       _logger->log(Logger::LOG_LEV_DEBUG, "Message succesfully serialized!");

@@ -48,7 +57,11 @@
    SerializableMessage* msg = NULL;
    try {
       _logger->log(Logger::LOG_LEV_VTRACE, "Deserializing message...");
+#ifdef TEXT_SERIALIZATION
+      boost::archive::text_iarchive archive(serializedMsg);
+#else
       boost::archive::binary_iarchive archive(serializedMsg);
+#endif
       archive >> BOOST_SERIALIZATION_NVP(msg);
       _logger->log(Logger::LOG_LEV_VTRACE, "Message succesfully
deserialized:");
       msg->log();
```

./lib/arblib/base/SerializableMessage.h

```
--- lib/arblib/base/SerializableMessage.h   (revision 3700)
+++ lib/arblib/base/SerializableMessage.h   (revision 3848)
@@ -3,8 +3,13 @@

 #include <string>

+#ifdef TEXT_SERIALIZATION
+#include <boost/archive/text_iarchive.hpp>
+#include <boost/archive/text_oarchive.hpp>
+#else
 #include <boost/archive/binary_iarchive.hpp>
 #include <boost/archive/binary_oarchive.hpp>
+#endif

 #include "base/msgcodes.h"
 #include "Logger.h"
```

./lib/arblib/wfsArb/WfsPTypes.h

```
--- lib/arblib/wfsArb/WfsPTypes.h   (revision 3700)
+++ lib/arblib/wfsArb/WfsPTypes.h   (revision 3848)
@@ -2,6 +2,7 @@
 #define WFSPTYPES_H_INCLUDE

 #include "WfsInterfaceDefines.h"
+#include "AONanWrapper.hpp"

 using namespace Arcetri;

@@ -20,8 +21,33 @@
 }

 // prepareAcquireRefParams
+#ifdef TEXT_SERIALIZATION
 template<class Archive>
 void serialize(Archive & ar, Wfs_Arbitrator::prepareAcquireRefParams& params,
const unsigned int /* version */) {
+    AONanWrapper<float> _SOCoords[2] = {params.SOCoords[0],
params.SOCoords[1]};
+    AONanWrapper<float> _ROCoords[2] = {params.ROCoords[0],
```

```
params.ROCoords[1]};
+    AONanWrapper<float> _Elevation(params.Elevation);
+    AONanWrapper<float> _RotAngle(params.RotAngle);
+    AONanWrapper<float> _GravAngle(params.GravAngle);
+    AONanWrapper<float> _Mag(params.Mag);
+    AONanWrapper<float> _Color(params.Color);
+
+   ar & params.Instr;
+   ar & params.AOMode;
+   ar & _SOCoords[0];
+   ar & _SOCoords[1];
+   ar & _ROCoords[0];
+   ar & _ROCoords[1];
+   ar & _Elevation;
+   ar & _RotAngle;
+   ar & _GravAngle;
+   ar & _Mag;
+   ar & _Color;
+}
+#else
+template<class Archive>
+void serialize(Archive & ar, Wfs_Arbitrator::prepareAcquireRefParams& params,
const unsigned int /* version */) {
+
    ar & params.Instr;
    ar & params.AOMode;
    ar & params.SOCoords[0];
@@ -34,12 +60,24 @@
    ar & params.Mag;
    ar & params.Color;
 }
-
+#endif



 // modifyAOparams
+#ifdef TEXT_SERIALIZATION
+template<class Archive>
+void serialize(Archive & ar, Wfs_Arbitrator::modifyAOparams& params, const
```

```
unsigned int /* version */) {
+    AONanWrapper<float> _freq(params.freq), _TTmod(params.TTmod);
+    ar & _freq;
+    ar & params.Binning;
+    ar & _TTmod;
+    ar & params.Fw1Pos;
+    ar & params.Fw2Pos;
+    ar & params.checkCameralens;
+}
+#else
 template<class Archive>
 void serialize(Archive & ar, Wfs_Arbitrator::modifyAOparams& params, const
unsigned int /* version */) {
     ar & params.freq;
@@ -49,8 +87,37 @@
     ar & params.Fw2Pos;
     ar & params.checkCameralens;
 }
+#endif

 // acquireRefResult
+#ifdef TEXT_SERIALIZATION
+template<class Archive>
+void serialize(Archive & ar, Wfs_Arbitrator::acquireRefResult& params, const
unsigned int /* version */) {
+
+   AONanWrapper<float> _deltaXY[2] = {params.deltaXY[0], params.deltaXY[1]};
+   AONanWrapper<float> _freq(params.freq), _TTmod(params.TTmod),
_starMag(params.starMag), _gain(params.gain);
+
+   ar & _deltaXY[0];
+   ar & _deltaXY[1];
+   ar & params.nModes;
+   ar & _freq;
+   ar & params.bin;
+   ar & _TTmod;
+   ar & params.Fw1Pos;
+   ar & params.Fw2Pos;
+   ar & params.TVframe;
+   ar & params.pupils;
+   ar & params.base;
```

```
+   ar & params.rec;
+   ar & params.filtering;
+   ar & _gain;
+   ar & params.decimation;
+   ar & _starMag;
+   ar & params.CLbase;
+   ar & params.CLrec;
+   ar & params.CLfreq;
+}
+#else
 template<class Archive>
 void serialize(Archive & ar, Wfs_Arbitrator::acquireRefResult& params, const
unsigned int /* version */) {

@@ -74,6 +141,7 @@
    ar & params.CLrec;
    ar & params.CLfreq;
 }
+#endif

 // getTVSnap
 template<class Archive>
@@ -84,9 +152,29 @@
 }

 // prepareAcquireRefResult
+#ifdef TEXT_SERIALIZATION
 template<class Archive>
 void serialize(Archive & ar, Wfs_Arbitrator::prepareAcquireRefResult& params,
const unsigned int /* version */) {
+
+   AONanWrapper<float> _freq(params.freq), _TTmod(params.TTmod),
_gain(params.gain);

+   ar & _freq;
+   ar & params.binning;
+   ar & _TTmod;
+   ar & params.Fw1Pos;
+   ar & params.Fw2Pos;
+   ar & _gain;
+   ar & params.pupils;
```

```
+   ar & params.base;
+   ar & params.rec;
+   ar & params.filtering;
+   ar & params.decimation;
+   ar & params.nModes;
+}
+#else
+template<class Archive>
+void serialize(Archive & ar, Wfs_Arbitrator::prepareAcquireRefResult& params,
const unsigned int /* version */) {
+
    ar & params.freq;
    ar & params.binning;
    ar & params.TTmod;
@@ -100,6 +188,7 @@
    ar & params.decimation;
    ar & params.nModes;
 }
+#endif

 // getTVSnapResult
 template<class Archive>
@@ -150,18 +239,36 @@
 }

 // offsetXYparams
+#ifdef TEXT_SERIALIZATION
+template<class Archive>
+void serialize(Archive & ar, Wfs_Arbitrator::offsetXYparams& params, const
unsigned int /*version*/) {
+    AONanWrapper<double> _offsetX(params.offsetX), _offsetY(params.offsetY);
+    ar & _offsetX;
+    ar & _offsetY;
+    ar & params.brake;
+}
+#else
 template<class Archive>
 void serialize(Archive & ar, Wfs_Arbitrator::offsetXYparams& params, const
unsigned int /*version*/) {
     ar & params.offsetX;
     ar & params.offsetY;
```

```
     ar & params.brake;
 }
+#endif

 // offsetZparams
+#ifdef TEXT_SERIALIZATION
+template<class Archive>
+void serialize(Archive & ar, Wfs_Arbitrator::offsetZparams& params, const
unsigned int /*version*/) {
+    AONanWrapper<double> _offsetZ(params.offsetZ);
+    ar & _offsetZ;
+}
+#else
 template<class Archive>
 void serialize(Archive & ar, Wfs_Arbitrator::offsetZparams& params, const
unsigned int /*version*/) {
     ar & params.offsetZ;
 }
+#endif

 // correctModes
 template<class Archive>
@@ -170,18 +277,37 @@
 }

 // checkRef
+#ifdef TEXT_SERIALIZATION
+template<class Archive>
+void serialize(Archive & ar, Wfs_Arbitrator::checkRefResult& params, const
unsigned int /*version*/) {
+    AONanWrapper<float> _deltaXY[2] = {params.deltaXY[0], params.deltaXY[1]};
+    AONanWrapper<float> _starMag(params.starMag);
+    ar & _deltaXY;
+    ar & _starMag;
+}
+#else
 template<class Archive>
 void serialize(Archive & ar, Wfs_Arbitrator::checkRefResult& params, const
unsigned int /*version*/) {
     ar & params.deltaXY;
     ar & params.starMag;
```

```
 }
+#endif

 // setSource
+#ifdef TEXT_SERIALIZATION
+template<class Archive>
+void serialize(Archive & ar, Wfs_Arbitrator::setSourceParams& params, const
unsigned int /*version*/) {
+    AONanWrapper<float> _magnitude(params.magnitude);
+    ar & params.source;
+    ar & _magnitude;
+}
+#else
 template<class Archive>
 void serialize(Archive & ar, Wfs_Arbitrator::setSourceParams& params, const
unsigned int /*version*/) {
     ar & params.source;
     ar & params.magnitude;
 }
+#endif

 // antiDrift
 template<class Archive>
```

Added files

The following files were added and can be directly retireved from the repository

```
./lib/AONanWrapper.hpp
./getserializationflags.py
./getboost.py
```

-- LucaFini - 08 Apr 2015

Edit | Attach | Print version | History: r4 < r3 < r2 < r1 | Backlinks | View wiki text | Edit wiki text | More topic actions

Topic revision: r4 - 09 Apr 2015, AlfioPuglisi