# MOOSLIB Reference Manual

Generated by Doxygen 1.4.6

Tue Mar 13 10:22:45 2007

# Contents

**1 MOOSLIB Hierarchical Index** **1**

1.1 MOOSLIB Class Hierarchy . . . . . 1

**2 MOOSLIB Class Index** **3**

2.1 MOOSLIB Class List . . . . . 3

**3 MOOSLIB Class Documentation** **5**

3.1 CMOOSApp Class Reference . . . . . 5

3.2 CMOOSCommClient Class Reference . . . . . 12

3.3 CMOOSCommObject Class Reference . . . . . 20

3.4 CMOOSCommPkt Class Reference . . . . . 21

3.5 CMOOSCommServer Class Reference . . . . . 23

3.6 CMOOSException Class Reference . . . . . 29

3.7 CMOOSInstrument Class Reference . . . . . 30

3.8 CMOOSMsg Class Reference . . . . . 33

# Chapter 1

# MOOSLIB Hierarchical Index

## 1.1 MOOSLIB Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:


CMOOSApp . . . . . 5
  CMOOSInstrument . . . . . 30
CMOOSCommObject . . . . . 20
  CMOOSCommClient . . . . . 12
  CMOOSCommServer . . . . . 23
CMOOSCommPkt . . . . . 21
CMOOSException . . . . . 29
CMOOSMsg . . . . . 33

# Chapter 2

# MOOSLIB Class Index

## 2.1 MOOSLIB Class List

Here are the classes, structs, unions and interfaces with brief descriptions:


**CMOOSApp** . . . . . 5
**CMOOSCommClient** . . . . . 12
**CMOOSCommObject** . . . . . 20
**CMOOSCommPkt** . . . . . 21
**CMOOSCommServer** . . . . . 23
**CMOOSException** . . . . . 29
**CMOOSInstrument** . . . . . 30
**CMOOSMsg** . . . . . 33

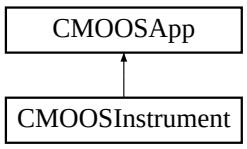
# Chapter 3

# MOOSLIB Class Documentation

## 3.1 CMOOSApp Class Reference

`#include <MOOSApp.h>`

Inheritance diagram for CMOOSApp::




### Public Member Functions

- bool **Run** (char *sName, char *sMissionFile)
- virtual bool **OnConnectToServer** ()
- virtual bool **OnDisconnectFromServer** ()

### Protected Member Functions

- virtual bool **Iterate** ()
- virtual bool **OnNewMail** (MOOSMSG_LIST &NewMail)
- void **SetCommsFreq** (unsigned int nFreq)
- void **SetAppFreq** (double dfFreq)
- double **GetAppStartTime** ()
- void **SetServer** (const char *sServerHost="LOCALHOST", long lPort=9000)
- bool **UseMOOSComms** (bool bUse)
- bool **MOOSDebugWrite** (const std::string &sTxt)
- std::string **GetAppName** ()
- std::string **GetMissionFileName** ()
- bool **AddMOOSVariable** (std::string sName, std::string sSubscribeName, std::string sPublishName, double dfCommsTime)
- CMOOSVariable * **GetMOOSVar** (std::string sName)
- bool **RegisterMOOSVariables** ()

- bool **UpdateMOOSVariables** (MOOSMSG_LIST &NewMail)
- bool **SetMOOSVar** (const std::string &sName, const std::string &sVal, double dfTime)
- bool **SetMOOSVar** (const std::string &sVarName, double dfVal, double dfTime)
- bool **PublishFreshMOOSVariables** ()
- bool **IsSimulateMode** ()
- virtual bool **OnStartUp** ()
- bool **ConfigureComms** ()
- double **GetTimeSinceIterate** ()
- double **GetLastIterateTime** ()
- int **GetIterateCount** ()
- bool **IsDebug** ()

## Protected Attributes

- **CMOOSCommClient m_Comms**
- CProcessConfigReader **m_MissionReader**
- MOOSVARMAP **m_MOOSVars**
- bool **m_bSimMode**
- long **m_lServerPort**
- std::string **m_sServerHost**
- std::string **m_sServerPort**
- bool **m_bServerSet**
- bool **m_bUseMOOSComms**
- std::string **m_sAppName**
- unsigned int **m_nCommsFreq**
- double **m_dfFreq**
- std::string **m_sMissionFile**
- double **m_dfAppStartTime**
- double **m_dfLastRunTime**
- bool **m_bDebug**

### 3.1.1 Detailed Description

This is a class from which all MOOS component applications can be derived main() will typically end with a call to MOOSAppDerivedClass::Run(). It provides automatic connection to the MOOSDB, provides slots for Mail Processing and application work, callbacks for connection/disconnection to MOOSDB, Configuration file reading and dynamic (runtime) variables. Definately worth getting to know.

### 3.1.2 Member Function Documentation

#### 3.1.2.1 bool CMOOSApp::AddMOOSVariable (std::string *sName*, std::string *sSubscribeName*, std::string *sPublishName*, double *dfCommsTime*) [protected]

Add a dynamic (run time) variable

**Parameters:**

***sName*** name of the variable

***sSubscribeName*** if you call **RegisterMOOSVariables()**(p. 8) the variable will be updated with mail called <sSubscribeName> if and when you call **UpdateMOOSVariables()**(p. 9)

***sPublishName*** if you call **PublishFreshMOOSVariables()**(p. 8) (and you've written to the dynamic varible since the last call) the variable will be published under this name.

***CommsTime*** - if sSubscribeName is not empty this is the minimum time between updates which you are interested in knowing about, so if CommsTime=0.1 then the maximum update rate you will see on the variable from the DB is 10HZ.

#### 3.1.2.2 bool CMOOSApp::ConfigureComms () [protected]

start up the comms

#### 3.1.2.3 string CMOOSApp::GetAppName () [protected]

return the application name

#### 3.1.2.4 double CMOOSApp::GetAppStartTime () [protected]

return the boot time of the App

#### 3.1.2.5 int CMOOSApp::GetIterateCount () [protected]

return number of times iterate has been called

#### 3.1.2.6 double CMOOSApp::GetLastIterateTime () [protected]

Return time at which the Run loop last ran (called Iterate)

#### 3.1.2.7 std::string CMOOSApp::GetMissionFileName () [protected]

return the application mission file name

#### 3.1.2.8 CMOOSVariable* CMOOSApp::GetMOOSVar (std::string *sName*) [protected]

return a pointer to a named variable

#### 3.1.2.9 double CMOOSApp::GetTimeSinceIterate () [protected]

Time since last iterate was called

#### 3.1.2.10 bool CMOOSApp::IsSimulateMode () [protected]

Returns true if Simulate = true is found in the mission/configuration file (a global flag) - the mission file is not re-read on each call

#### 3.1.2.11 bool CMOOSApp::Iterate () [protected, virtual]

called when the application should iterate. Overload this function in a derived class and within it write all the application specific code. It will be called at approximately nFreq = 1/AppTick Hz

#### 3.1.2.12 bool CMOOSApp::MOOSDebugWrite (const std::string & *sTxt*) [protected]

Call this to write a debug string to the DB under the name "MOOS_DEBUG"

#### 3.1.2.13 bool CMOOSApp::OnConnectToServer () [virtual]

Called when the class has succesully connected to the server. Overload this function and place use it to register for notification when variables of interest change

#### 3.1.2.14 bool CMOOSApp::OnDisconnectFromServer () [virtual]

Called when the class has disconnects from the server. Put code you want to run when this happens in a virtual version of this method

#### 3.1.2.15 bool CMOOSApp::OnNewMail (MOOSMSG_LIST & *NewMail*) [protected, virtual]

called when new mail has arrived. Overload this method in a derived class to process new mail. It will be called at approximately 1/CommsTick Hz. In this function you'll most likely interate over the collection of mail message received or call a m_Comms::PeekMail() to look for a specific named message.

**Parameters:**

***NewMail*** a list of new mail messages

#### 3.1.2.16 bool CMOOSApp::OnStartUp () [protected, virtual]

called just before the main app loop is entered. Specific initialisation code can be written in an overloaded version of this function

Reimplemented in **CMOOSInstrument** (p. 31).

#### 3.1.2.17 bool CMOOSApp::PublishFreshMOOSVariables () [protected]

Send any variables (under their sPublishName see AddMOOSVariable) which been written too since the last call of **PublishFreshMOOSVariables()**(p. 8)

#### 3.1.2.18 bool CMOOSApp::RegisterMOOSVariables () [protected]

Register with the DB to be mailed about any changes to any dynamic variables which were created with non-empty sSubscribeName fields

#### 3.1.2.19 bool CMOOSApp::Run (char * *sName*, char * *sMissionFile*)

called to start the application

**Parameters:**

***sName*** The name of this application (must be unique amoungst MOOS components

***the*** name of the mission file

#### 3.1.2.20 void CMOOSApp::SetAppFreq (double *dfFreq*) [protected]

Set the time between calls of Iterate (which is where you'll probably do Application work)- can be set using the AppTick flag in the config file

#### 3.1.2.21 void CMOOSApp::SetCommsFreq (unsigned int *nFreq*) [protected]

Set the time between calls into the DB - can be set using the CommsTick flag in the config file

#### 3.1.2.22 bool CMOOSApp::SetMOOSVar (const std::string & *sVarName*, double *dfVal*, double *dfTime*) [protected]

Set value in a dynamic variable if the variable is of type string (type is set on first write )

#### 3.1.2.23 bool CMOOSApp::SetMOOSVar (const std::string & *sName*, const std::string & *sVal*, double *dfTime*) [protected]

Set value in a dynamic variable if teh variable is of type double (type is set on first write )

#### 3.1.2.24 void CMOOSApp::SetServer (const char * *sServerHost* = "LOCALHOST", long *lPort* = 9000) [protected]

Called to set the MOOS server info used rarely usually this info will be picked up by the MOOSApp automatically when it Run is called specifying the configuration file (which contains the DB's coordinates)

**Parameters:**

***sServerHost*** name of the machine hosting the MOOSDB application

***lPort*** port nuymber that MOOSDB listens on

#### 3.1.2.25 bool CMOOSApp::UpdateMOOSVariables (MOOSMSG_LIST & *NewMail*) [protected]

Pass mail (usually collected in OnNewMail) to the set of dynamic variables. If they are interested (mail name matches their subscribe name) they will update themselves automatically

#### 3.1.2.26 bool CMOOSApp::UseMOOSComms (bool *bUse*) [protected]

By default MOOSDB comms are on - but you may want to use the structuire of MOOSApp as a standalone application - if so call this function with a false parameter

### 3.1.3 Member Data Documentation

#### 3.1.3.1 bool CMOOSApp::m_bServerSet [protected]

true if the server has been set

#### 3.1.3.2 bool CMOOSApp::m_bSimMode [protected]

flag saying whether MOOS is running with a simulator can be set by registering for SIMULATION_MODE variable

#### 3.1.3.3 bool CMOOSApp::m_bUseMOOSComms [protected]

true if we want to use MOOS comms

#### 3.1.3.4 CMOOSCommClient CMOOSApp::m_Comms [protected]

The MOOSComms node. All communications happens by way of this object. You'll often do things like m_Comms.Notify("VARIABLE_X","STRING_DATA",dfTime) top send data

#### 3.1.3.5 double CMOOSApp::m_dfAppStartTime [protected]

The start time of the application

#### 3.1.3.6 double CMOOSApp::m_dfFreq [protected]

frequency at which this application will iterate

#### 3.1.3.7 double CMOOSApp::m_dfLastRunTime [protected]

Time at which the Run loop last ran (called Iterate)

#### 3.1.3.8 long CMOOSApp::m_lServerPort [protected]

Port on which server application listens for new connection

#### 3.1.3.9 CProcessConfigReader CMOOSApp::m_MissionReader [protected]

a very useful object that lets us retrieve configuration information from the mission file using calls like GetConfigurationParam()

#### 3.1.3.10 MOOSVARMAP CMOOSApp::m_MOOSVars [protected]

a map of dynamic/run time moos variables that may be set by comms - avoid messy long if else if statements

#### 3.1.3.11 unsigned int CMOOSApp::m_nCommsFreq [protected]

frequency at which server will be contacted

#### 3.1.3.12 std::string CMOOSApp::m_sAppName [protected]

name of this application

#### 3.1.3.13 std::string CMOOSApp::m_sMissionFile [protected]

std::string name of mission file

#### 3.1.3.14 std::string CMOOSApp::m_sServerHost [protected]

name of machine on which MOOS Server resides

#### 3.1.3.15 std::string CMOOSApp::m_sServerPort [protected]

std::string version of m_lServerPort

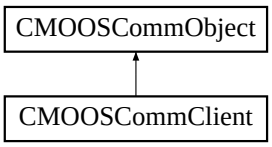
The documentation for this class was generated from the following files:

- /home/pnewman/code/MOOS/trunk/Core/MOOSLIB/MOOSApp.h
- /home/pnewman/code/MOOS/trunk/Core/MOOSLIB/MOOSApp.cpp

## 3.2 CMOOSCommClient Class Reference

`#include <MOOSCommClient.h>`

Inheritance diagram for CMOOSCommClient::




### Public Member Functions

- **CMOOSCommClient** ()
  *default constructor*
- virtual ~**CMOOSCommClient** ()
  *default destructor*
- bool **Notify** (const std::string &sVar, const std::string &sVal, double dfTime=-1)
- bool **Notify** (const std::string &sVar, double dfVal, double dfTime=-1)
- bool **Register** (const std::string &sVar, double dfInterval)
- bool **UnRegister** (const std::string &sVar)
- bool **IsConnected** ()
- bool **Fetch** (MOOSMSG_LIST &MsgList)
- bool **Post** (**CMOOSMsg** &Msg)
- bool **ClientLoop** ()
- virtual bool **DoClientWork** ()
- bool **Run** (const char *sServer, long lPort, const char *sMyName, unsigned int nFundamentalFreq=5)
- void **SetOnConnectCallBack** (bool(*pfn)(void *pParamCaller), void *pCallerParam)
- void **SetOnDisconnectCallBack** (bool(*pfn)(void *pParamCaller), void *pCallerParam)
- bool **ServerRequest** (const std::string &sWhat, MOOSMSG_LIST &MsgList, double dfTimeOut=2.0, bool bContinuouslyClearBox=true)
- bool **Peek** (MOOSMSG_LIST &List, int nIDRequired, bool bClearBox=true)
- std::string **GetDescription** ()
- bool **FakeSource** (bool bFake)
- bool **Close** (bool bNice=true)

### Static Public Member Functions

- static bool **PeekMail** (MOOSMSG_LIST &Mail, const std::string &sKey, **CMOOSMsg** &Msg, bool bErase=false, bool bFindYoungest=false)
- static std::string **GetLocalIPAddress** ()

### Protected Types

- typedef pthread_t **THREAD_ID**

## Protected Member Functions

- bool **ClearResources** ()
- void **DoBanner** ()
- bool **OnCloseConnection** ()
- bool **HandShake** ()
- bool **ConnectToServer** ()
- bool **StartThreads** ()

## Protected Attributes

- int **m_nNextMsgID**
- bool **m_bConnected**
- bool **m_bFakeSource**
- unsigned int **m_nOutPendingLimit**
- unsigned int **m_nInPendingLimit**
- std::string **m_sMyName**
- CMOOSLock **m_OutLock**
- CMOOSLock **m_InLock**
- XPCTcpSocket * **m_pSocket**
- std::string **m_sDBHost**
- long **m_lPort**
- bool **m_bQuit**
- bool **m_bMailPresent**
- **THREAD_ID m_nClientThreadID**
- MOOSMSG_LIST **m_OutBox**
- MOOSMSG_LIST **m_InBox**
- void * **m_pConnectCallBackParam**
- bool(* **m_pfnConnectCallBack** )(void *pConnectParam)
- void * **m_pDisconnectCallBackParam**
- bool(* **m_pfnDisconnectCallBack** )(void *pParam)
- unsigned int **m_nFundamentalFreq**
- std::set< std::string > **m_Registered**

### 3.2.1 Detailed Description

This class is the most important component of MOOS as seen from the eyes of a component developer

### 3.2.2 Member Typedef Documentation

#### 3.2.2.1 typedef pthread_t CMOOSCommClient::THREAD_ID [protected]

ID of IO thread

### 3.2.3 Member Function Documentation

#### 3.2.3.1 bool CMOOSCommClient::ClientLoop ()

internal method which runs in a seperate thread and manges the input and output of messages from thser server. DO NOT CALL THIS METHOD.

#### 3.2.3.2 bool CMOOSCommClient::Close (bool *bNice* = `true`)

make the client shut down

#### 3.2.3.3 bool CMOOSCommClient::ConnectToServer () `[protected]`

Connect to the server process using info supplied to Init

**See also:**
Init

#### 3.2.3.4 void CMOOSCommClient::DoBanner () `[protected]`

send library info to stdout

#### 3.2.3.5 bool CMOOSCommClient::DoClientWork () `[virtual]`

called by the above to do the client mail box shuffling

#### 3.2.3.6 bool CMOOSCommClient::FakeSource (bool *bFake*)

call with "true" if you want thsi client to fake its own outgoing name. This is rarely used but very useful when it is

#### 3.2.3.7 bool CMOOSCommClient::Fetch (MOOSMSG_LIST & *MsgList*)

Called by a user of **CMOOSCommClient**(p. 12) to retrieve mail

**Parameters:**
***MsgList*** a list of messages into which the newly received message will be placed

**Returns:**
true if there is new mail

#### 3.2.3.8 string CMOOSCommClient::GetDescription ()

describe this client in a string

#### 3.2.3.9 string CMOOSCommClient::GetLocalIPAddress () `[static]`

return a string of the host machines's IP adress

#### 3.2.3.10 bool CMOOSCommClient::HandShake () `[protected]`

performs a handshake with the server when a new connection is made. Within this function this class tells teh server its name

#### 3.2.3.11 bool CMOOSCommClient::IsConnected ()

returns true if this obecjt is connected to the server

#### 3.2.3.12 bool CMOOSCommClient::Notify (const std::string & *sVar*, double *dfVal*, double *dfTime* = -1)

notify the MOOS community that something has changed (double)

#### 3.2.3.13 bool CMOOSCommClient::Notify (const std::string & *sVar*, const std::string & *sVal*, double *dfTime* = -1)

notify the MOOS community that something has changed (string)

#### 3.2.3.14 bool CMOOSCommClient::OnCloseConnection () [protected]

called when connection to server is closed

#### 3.2.3.15 bool CMOOSCommClient::Peek (MOOSMSG_LIST & *List*, int *nIDRequired*, bool *bClearBox* = true)

Have a peek at mail box and remove a particular message, by default all other messages are removed. Note this is quite different from PeekMail

#### 3.2.3.16 static bool CMOOSCommClient::PeekMail (MOOSMSG_LIST & *Mail*, const std::string & *sKey*, CMOOSMsg & *Msg*, bool *bErase* = false, bool *bFindYoungest* = false) [static]

a static helper function hat lets a user browse a mail list the message is removed if bremove is true

#### 3.2.3.17 bool CMOOSCommClient::Post (CMOOSMsg & *Msg*)

place a single message in the out box and return immediately. Completion of this method does not infer transmission. However transmission will occur at the next available oppurtunity. Inpractice the apparent speed of message transmission will be very fast indeed. This model howver prevents wayward user software bring down the MOOSComms by way of denial of service. (ie hogging the network)

**Parameters:**

***Msg*** reference to **CMOOSMsg**(p. 33) which user whishes to send

#### 3.2.3.18 bool CMOOSCommClient::Register (const std::string & *sVar*, double *dfInterval*)

Register for notification in changes of named variable

**Parameters:**

***sVar*** name of variable of interest

***dfInterval*** minimum time between notifications

#### 3.2.3.19 bool CMOOSCommClient::Run (const char * *sServer*, long *lPort*, const char * *sMyName*, unsigned int *nFundamentalFreq* = 5)

Run the MOOSCommClient Object. This call is non blocking and begins managing process IO with the MOOSComms protocol

**Parameters:**

***sServer*** Name of machine on which server resides eg LOCALHOST or guru.mit.edu

***lPort*** number of port on which server is listening for new connections eg 9000

***sMyName*** std::string name by which this MOOS process will be known - eg "MotionController" or "DepthSensor"

***nFundamentalFrequency*** the basic tick frequency of the comms loop. Default value of 5 implies mail will be retrieved and sent from the server at 5Hz

#### 3.2.3.20 bool CMOOSCommClient::ServerRequest (const std::string & *sWhat*, MOOSMSG_LIST & *MsgList*, double *dfTimeOut* = 2.0, bool *bContinuouslyClearBox* = true)

Directly and asynhrounously make a request to the server (use this very rarely if ever. Its not meant for public consumption)

**Parameters:**

***sWhat*** string specifying what request to make - ALL, DB_CLEAR, PROCESS_SUMMARY or VAR_SUMMARY

***MsgList*** List of messages returned by server

***dfTimeOut*** TimeOut

***bContinuouslyClearBox*** true if all other message returned with query are to be discarded.

#### 3.2.3.21 void CMOOSCommClient::SetOnConnectCallBack (bool(*)(void *pParamCaller) *pfn*, void * *pCallerParam*)

set the user supplied OnConnect call back. This callback , when set, will be invoked when a connection to the server is made. It is a good plan to register for notification of variables in this callback

**Parameters:**

***pfn*** pointer to static function of type bool Fn(void * pParam)

***pCallerParam*** parameter passed to callback function when invoked

#### 3.2.3.22 void CMOOSCommClient::SetOnDisconnectCallBack (bool(*)(void *pParamCaller) *pfn*, void * *pCallerParam*)

set the user supplied OnConnect call back. This callback , when set, will be invoked when a connection to the server is lost.

**Parameters:**

***pfn*** pointer to static function of type bool Fn(void * pParam)

***pCallerParam*** parameter passed to callback function when invoked

#### 3.2.3.23 bool CMOOSCommClient::StartThreads () [protected]

called internally to start IO management thread

**See also:**
**ClientLoop**(p. 13)

#### 3.2.3.24 bool CMOOSCommClient::UnRegister (const std::string & *sVar*)

UnRegister for notification in changes of named variable

**Parameters:**
***sVar*** name of variable of interest

### 3.2.4 Member Data Documentation

#### 3.2.4.1 bool CMOOSCommClient::m_bConnected [protected]

true if we are connected to the server

#### 3.2.4.2 bool CMOOSCommClient::m_bFakeSource [protected]

true if we want to be able to fake sources of messages (used by playback)

#### 3.2.4.3 bool CMOOSCommClient::m_bMailPresent [protected]

true if mail present (saves using a semaphore to open an empty box)

#### 3.2.4.4 bool CMOOSCommClient::m_bQuit [protected]

IO thread will continue so long as this flag is false

#### 3.2.4.5 MOOSMSG_LIST CMOOSCommClient::m_InBox [protected]

List of message that have been received and are ready for reading by user

**See also:**
**Fetch**(p. 14)

#### 3.2.4.6 CMOOSLock CMOOSCommClient::m_InLock [protected]

Mutex around incoming mail box

**See also:**
CMOOSLock

#### 3.2.4.7 long CMOOSCommClient::m_lPort [protected]

port number on which server process is listening for new connections

**See also:**
Init

#### 3.2.4.8 unsigned int CMOOSCommClient::m_nFundamentalFreq [protected]

funcdamental frequency with which comms with server occurs

**See also:**
**Run**(p. 16)

#### 3.2.4.9 unsigned int CMOOSCommClient::m_nInPendingLimit [protected]

The number of unread incoming messages that can be tolerated

#### 3.2.4.10 unsigned int CMOOSCommClient::m_nOutPendingLimit [protected]

The number of pending unsent messages that can be tolerated

#### 3.2.4.11 MOOSMSG_LIST CMOOSCommClient::m_OutBox [protected]

List of messages that a pending to be sent

**See also:**
**Post**(p. 15)

#### 3.2.4.12 CMOOSLock CMOOSCommClient::m_OutLock [protected]

Mutex around Outgoing mail box

**See also:**
CMOOSLock

#### 3.2.4.13 void* CMOOSCommClient::m_pConnectCallBackParam [protected]

parameter that user wants passed to him/her with connect callback

#### 3.2.4.14 void* CMOOSCommClient::m_pDisconnectCallBackParam [protected]

parameter that user wants passed to him/her with disconnect callback

#### 3.2.4.15 bool(* CMOOSCommClient::m_pfnConnectCallBack)(void *pConnectParam) [protected]

the user supplied OnConnect callback

#### 3.2.4.16 bool(* CMOOSCommClient::m_pfnDisconnectCallBack)(void *pParam) [protected]

the user supplied OnDisConnect callback

#### 3.2.4.17 XPCTcpSocket* CMOOSCommClient::m_pSocket [protected]

pointer to socket connected to server

#### 3.2.4.18 std::set<std::string> CMOOSCommClient::m_Registered [protected]

a set of strings of the resources that have been registered for

#### 3.2.4.19 std::string CMOOSCommClient::m_sDBHost [protected]

name of teh host on which the server process lives

**See also:**
Init

#### 3.2.4.20 std::string CMOOSCommClient::m_sMyName [protected]

name of MOOS process

**See also:**
Init

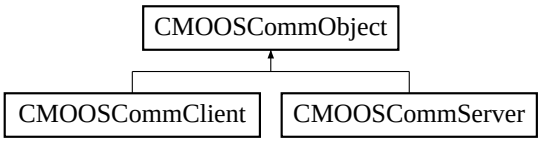
The documentation for this class was generated from the following files:

- /home/pnewman/code/MOOS/trunk/Core/MOOSLIB/MOOSCommClient.h
- /home/pnewman/code/MOOS/trunk/Core/MOOSLIB/MOOSCommClient.cpp

## 3.3 CMOOSCommObject Class Reference

`#include <MOOSCommObject.h>`

Inheritance diagram for CMOOSCommObject::




### Static Public Member Functions

- static bool **SocketsInit** ()

  *called to intialise system socket services. Only does something useful in Win32 land*

### Protected Member Functions

- bool **SendPkt** (XPCTcpSocket *pSocket, **CMOOSCommPkt** &PktTx)
- bool **ReadPkt** (XPCTcpSocket *pSocket, **CMOOSCommPkt** &PktRx, int nSecondsTimeOut=-1)
- bool **SendMsg** (XPCTcpSocket *pSocket, **CMOOSMsg** &Msg)
- bool **ReadMsg** (XPCTcpSocket *pSocket, **CMOOSMsg** &Msg, int nSecondsTimeOut=-1)

### 3.3.1 Detailed Description

A base class for the **CMOOSCommServer**(p. 23) and **CMOOSCommClient**(p. 12) objects. This class provides basic Receive and Transmit capabilities of CMOOSMsg's and CMOOSCommPkts. Where messages are passed as parameters then there are transparently packed into packets.

The documentation for this class was generated from the following files:

- /home/pnewman/code/MOOS/trunk/Core/MOOSLIB/MOOSCommObject.h
- /home/pnewman/code/MOOS/trunk/Core/MOOSLIB/MOOSCommObject.cpp

## 3.4 CMOOSCommPkt Class Reference

`#include <MOOSCommPkt.h>`

### Public Member Functions

- bool **Serialize** (MOOSMSG_LIST &List, bool bToStream=true, bool bNoNULL=false)
- int **GetStreamLength** ()
- bool **Fill** (unsigned char *InData, int nData)
- int **GetBytesRequired** ()

### Public Attributes

- unsigned char * **m_pStream**
- unsigned char * **m_pNextData**
- int **m_nStreamSpace**
- unsigned char **DefaultStream** [MOOS_PKT_DEFAULT_SPACE]

### Protected Member Functions

- bool **InflateTo** (int nNewStreamSize)
- bool **CopyToStream** (unsigned char *pData, int nBytes)

### Protected Attributes

- int **m_nByteCount**
- int **m_nMsgLen**
- bool **m_bAllocated**

### 3.4.1 Detailed Description

This class is part of MOOS's internal transport mechanism. It any number of CMOOSMsg's can be packed into a **CMOOSCommPkt**(p. 21) and sent in one lump between a **CMOOSCommServer**(p. 23) and **CMOOSCommClient**(p. 12) object. It is never used by a user of MOOSLib

### 3.4.2 Member Function Documentation

#### 3.4.2.1 bool CMOOSCommPkt::Serialize (MOOSMSG_LIST & *List*, bool *bToStream* = `true`, bool *bNoNULL* = `false`)

This function stuffs messages in/from a packet

### 3.4.3 Member Data Documentation

#### 3.4.3.1 bool CMOOSCommPkt::m_bAllocated `[protected]`

true is the packet has been infated to increase capicity and m_pStream no longer points to DefaultStream but to heap space allocated with new

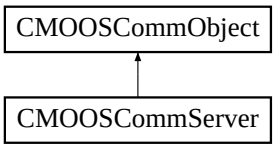
The documentation for this class was generated from the following files:

- /home/pnewman/code/MOOS/trunk/Core/MOOSLIB/MOOSCommPkt.h
- /home/pnewman/code/MOOS/trunk/Core/MOOSLIB/MOOSCommPkt.cpp

## 3.5 CMOOSCommServer Class Reference

`#include <MOOSCommServer.h>`

Inheritance diagram for CMOOSCommServer::




### Public Types

- typedef std::list< XPCTcpSocket * > **SOCKETLIST**
- typedef std::map< int, std::string > **SOCKETFD_2_CLIENT_NAME_MAP**
- typedef std::list< std::string > **STRING_LIST**

### Public Member Functions

- bool **GetClientNames** (STRING_LIST &sList)
- void **SetOnRxCallBack** (bool(*pfn)(const std::string &sClient, MOOSMSG_LIST &MsgListRx, MOOSMSG_LIST &MsgListTx, void *pParam), void *pParam)
- void **SetOnDisconnectCallBack** (bool(*pfn)(std::string &sClient, void *pParam), void *pParam)
- bool **ListenLoop** ()
- bool **ServerLoop** ()
- bool **TimerLoop** ()
- bool **Run** (long lPort, const std::string &sCommunityName)
- **CMOOSCommServer** ()

  *default constructor*

- virtual ~**CMOOSCommServer** ()

  *default destructor*

### Protected Types

- typedef pthread_t **THREAD_ID**

### Protected Member Functions

- int **GetMaxSocketFD** ()
- virtual void **DoBanner** ()
- std::string **GetClientName** (XPCTcpSocket *pSocket)
- void **PoisonClient** (XPCTcpSocket *pSocket, char *sReason)
- bool **HandShake** (XPCTcpSocket *pNewSocket)
- bool **IsUniqueName** (std::string &sClientName)
- virtual bool **OnClientDisconnect** ()

*called when a client disconnects or and error occurs*

- virtual bool **OnAbsentClient** (XPCTcpSocket *pClient)

  *called when a client goes quiet...*

- virtual bool **OnNewClient** (XPCTcpSocket *pNewClient, char *sName)
- virtual bool **ProcessClient** ()
- bool **StartThreads** ()

  *called from init to start teh listen and server threads up*

## Protected Attributes

- CMOOSLock **m_SocketListLock**
- int **m_nTotalActions**

  *internal count of the number of calls processed*

- **THREAD_ID m_nListenThreadID**

  *ID of Listen Thread.*

- **THREAD_ID m_nServerThreadID**

  *ID of Server Thread.*

- **THREAD_ID m_nTimerThreadID**

  *ID of timer Thread.*

- bool(* **m_pfnRxCallBack** )(const std::string &sClient, MOOSMSG_LIST &MsgListRx, MOOSMSG_LIST &MsgListTx, void *pCaller)
- void * **m_pRxCallBackParam**
- bool(* **m_pfnDisconnectCallBack** )(std::string &sClient, void *pParam)
- void * **m_pDisconnectCallBackParam**
- XPCTcpSocket * **m_pListenSocket**
- XPCTcpSocket * **m_pFocusSocket**
- SOCKETLIST **m_ClientSocketList**
- SOCKETFD_2_CLIENT_NAME_MAP **m_Socket2ClientMap**
- long **m_lListenPort**

  *port listen socket is bound to*

- bool **m_bQuit**

  *threads continue while this flag is false*

- int **m_nMaxSocketFD**

  *largest FD of all connected sockets*

- std::string **m_sCommunityName**

  *name of community being served*

### 3.5.1 Detailed Description

This class is the MOOS Comms Server. It lies at the heart of the communications architecture and typically is of no interest to the component developer. It maintains a list of all the connected clients and their names. It simultaneously listens on all sockets for calling clients and then calls a user supplied call back to handle the request. This class is only used by the CMOOSDB application

### 3.5.2 Member Typedef Documentation

#### 3.5.2.1 typedef pthread_t CMOOSCommServer::THREAD_ID [protected]

Win32 handle to Server thread

**See also:**
**ServerLoop**(p. 26)

### 3.5.3 Member Function Documentation

#### 3.5.3.1 void CMOOSCommServer::DoBanner () [protected, virtual]

prints class information banner to stdout

#### 3.5.3.2 string CMOOSCommServer::GetClientName (XPCTcpSocket * *pSocket*) [protected]

Get the name of the client on the remote end of pSocket

#### 3.5.3.3 int CMOOSCommServer::GetMaxSocketFD () [protected]

figures out what the largest socket FD of all connected sockets. (needed by select)

#### 3.5.3.4 bool CMOOSCommServer::HandShake (XPCTcpSocket * *pNewSocket*) [protected]

Perform handshaling with client just after a connection has been accepted

#### 3.5.3.5 bool CMOOSCommServer::IsUniqueName (std::string & *sClientName*) [protected]

returns true if a server has no connection to the named client

**Parameters:**
***sClientName*** reference to client name std::string

#### 3.5.3.6 bool CMOOSCommServer::ListenLoop ()

This function is the listen loop called from one of the two server threads. It is responsible for accepting a coonection and creating a new client socket.

#### 3.5.3.7 bool CMOOSCommServer::OnNewClient (XPCTcpSocket * *pNewClient*, char * *sName*) [protected, virtual]

Called when a new client connects. Performs handshaking and adds new socket to m_ClientSocketList

**Parameters:**

***pNewClient*** pointer to teh new socket created in ListenLoop;

**See also:**

**ListenLoop**(p. 25)

#### 3.5.3.8 void CMOOSCommServer::PoisonClient (XPCTcpSocket * *pSocket*, char * *sReason*) [protected]

Send a Poisoned mesasge to the client on the end of pSocket. This may cause teh client comms thrad to die

#### 3.5.3.9 bool CMOOSCommServer::ProcessClient () [protected, virtual]

called from Server loop this function handles all the processing for the current client call. It inturn invokes the user supplied callback function

#### 3.5.3.10 bool CMOOSCommServer::Run (long *lPort*, const std::string & *sCommunityName*)

Initialise the server. This is a non blocking call and launches the MOOS Comms server threads.

**Parameters:**

***lPort*** port number to listen on

#### 3.5.3.11 bool CMOOSCommServer::ServerLoop ()

This function is the server loop called from one of the two server threads. It listens to all presently connected sockets and when a call is received invokes thse user supplied callback

#### 3.5.3.12 void CMOOSCommServer::SetOnDisconnectCallBack (bool(*)(std::string &sClient, void *pParam) *pfn*, void * *pParam*)

Set the disconnect message call back handler. The supplied call back must be of the form static bool MyCallBack(std::string & sClient,, void * pParam).

**Parameters:**

***sClient*** contains the incoming messages.

***TxLst*** passed to the handler as a recepticle for all the message that should be sent back to the client in response to teh incoming messages.

***pParam*** user suplied parameter to be passed to callback function

#### 3.5.3.13 void CMOOSCommServer::SetOnRxCallBack (bool(∗)(const std::string &sClient, MOOSMSG_LIST &MsgListRx, MOOSMSG_LIST &MsgListTx, void ∗pParam) *pfn*, void ∗ *pParam*)

Set the recieve message call back handler. The callback will be called whenever a client sends one or more messages to teh server. The supplied call back must be of the form static bool MyCallBack(MOOSMSG_LIST & RxLst,MOOSMSG_LIST & TxLst, void ∗ pParam).

**Parameters:**

***sClient*** Name of client at the end of the socket sending this Pkt

***RxLst*** contains the incoming messages.

***TxLst*** passed to the handler as a recepticle for all the message that should be sent back to the client in response to teh incoming messages.

***pParam*** user suplied parameter to be passed to callback function

#### 3.5.3.14 bool CMOOSCommServer::TimerLoop ()

This function is the timer loop called from one of the three server threads. It makes sure all clients speak occasionally

### 3.5.4 Member Data Documentation

#### 3.5.4.1 SOCKETLIST CMOOSCommServer::m_ClientSocketList [protected]

list of all currently connected sockets

#### 3.5.4.2 void∗ CMOOSCommServer::m_pDisconnectCallBackParam [protected]

place holder for teh address of the object passed back to the user during a Disconnect callback

**See also:**

**SetOnDisconnectCallBack**(p. 26)

#### 3.5.4.3 bool(∗ CMOOSCommServer::m_pfnDisconnectCallBack)(std::string &sClient, void ∗pParam) [protected]

user supplied OnDisconnect callback

**See also:**

**SetOnDisconnectCallBack**(p. 26)

#### 3.5.4.4 bool(∗ CMOOSCommServer::m_pfnRxCallBack)(const std::string &sClient, MOOSMSG_LIST &MsgListRx, MOOSMSG_LIST &MsgListTx, void ∗pCaller) [protected]

user supplied OnRx callback

**See also:**

**SetOnRxCallBack**(p. 27)

#### 3.5.4.5 XPCTcpSocket* CMOOSCommServer::m_pFocusSocket [protected]

pointer to the socket which server is currently processing call from

#### 3.5.4.6 XPCTcpSocket* CMOOSCommServer::m_pListenSocket [protected]

Listen socket (bound to port address supplied in constructor)

#### 3.5.4.7 void* CMOOSCommServer::m_pRxCallBackParam [protected]

place holder for teh address of the object passed back to the user during an Rx callback

**See also:**
**SetOnRxCallBack**(p. 27)

#### 3.5.4.8 SOCKETFD_2_CLIENT_NAME_MAP CMOOSCommServer::m_Socket2ClientMap [protected]

map of socket file descriptors to the std::string name of the client process at teh other end

#### 3.5.4.9 CMOOSLock CMOOSCommServer::m_SocketListLock [protected]

a simple mutex to guard access to m_ClientSocketList

**See also:**
**m_ClientSocketList**(p. 27)

The documentation for this class was generated from the following files:

- /home/pnewman/code/MOOS/trunk/Core/MOOSLIB/MOOSCommServer.h
- /home/pnewman/code/MOOS/trunk/Core/MOOSLIB/MOOSCommServer.cpp

## 3.6 CMOOSException Class Reference

`#include <MOOSException.h>`

### Public Member Functions

- **CMOOSException** (const char *sStr)
- **CMOOSException** (const std::string &s)
- char * **c_str** ()

### Public Attributes

- char **m_sReason** [100]

  *storage for the exception reason*

### 3.6.1 Detailed Description

A trivial Exception class

### 3.6.2 Constructor & Destructor Documentation

#### 3.6.2.1 CMOOSException::CMOOSException (const char * *sStr*)

construct an exception with a string argument giving the reason for teh exception

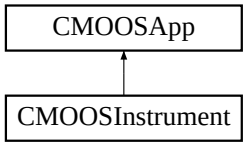
The documentation for this class was generated from the following files:

- /home/pnewman/code/MOOS/trunk/Core/MOOSLIB/MOOSException.h
- /home/pnewman/code/MOOS/trunk/Core/MOOSLIB/MOOSException.cpp

## 3.7 CMOOSInstrument Class Reference

`#include <MOOSInstrument.h>`

Inheritance diagram for CMOOSInstrument::




### Static Public Member Functions

- static std::string **Message2NMEA** (std::string sMsg)
- static bool **DoNMEACheckSum** (std::string sNMEA)

### Public Attributes

- CMOOSLinuxSerialPort **m_Port**

### Protected Member Functions

- virtual bool **OnStartUp** ()
- virtual bool **InitialiseSensor** ()
- bool **InitialiseSensorN** (int nAttempts, std::string sSensorName)
- virtual bool **SetupPort** ()
- bool **PublishRaw** ()
- double **GetMagneticOffset** ()
- void **SetInstrumentErrorMessage** (std::string sError)
- void **SetPrompt** (std::string sPrompt)

### Protected Attributes

- bool **m_bPublishRaw**
- std::string **m_sResourceName**
- double **m_dfMagneticOffset**
- std::string **m_sPrompt**
- std::string **m_sInstrumentErrorMessage**

### 3.7.1 Detailed Description

Class that derives from **CMOOSApp**(p. 5) and adds functionality of cross platform serial ports

## 3.7.2 Member Function Documentation

### 3.7.2.1 static bool CMOOSInstrument::DoNMEACheckSum (std::string *sNMEA*) [static]

performs NMEA string checksum

### 3.7.2.2 double CMOOSInstrument::GetMagneticOffset () [protected]

some legacy stuff that should be removed...

### 3.7.2.3 bool CMOOSInstrument::InitialiseSensor () [protected, virtual]

called from OnStartUp - overload to execute custom start up code for sensor

### 3.7.2.4 bool CMOOSInstrument::InitialiseSensorN (int *nAttempts*, std::string *sSensorName*) [protected]

called from OnStartUp class InitialiseSensor N times

### 3.7.2.5 static std::string CMOOSInstrument::Message2NMEA (std::string *sMsg*) [static]

turns a string into NMEA string

### 3.7.2.6 bool CMOOSInstrument::OnStartUp () [protected, virtual]

**CMOOSApp**(p. 5) overide
Reimplemented from **CMOOSApp** (p. 8).

### 3.7.2.7 bool CMOOSInstrument::PublishRaw () [inline, protected]

returns true if instrument is publishing raw data

### 3.7.2.8 bool CMOOSInstrument::SetupPort () [protected, virtual]

Set up the serial port by reading paremters from mission file

## 3.7.3 Member Data Documentation

### 3.7.3.1 bool CMOOSInstrument::m_bPublishRaw [protected]

set to true if this instrument should publish the raw incoming data to the DB

### 3.7.3.2 double CMOOSInstrument::m_dfMagneticOffset [protected]

some legacy stuff that should be removed...

#### 3.7.3.3 CMOOSLinuxSerialPort CMOOSInstrument::m_Port

A sensor port

#### 3.7.3.4 std::string CMOOSInstrument::m_sResourceName [protected]

a place holder for a the name of this sensor resource - rarely used

The documentation for this class was generated from the following files:

- /home/pnewman/code/MOOS/trunk/Core/MOOSLIB/MOOSInstrument.h
- /home/pnewman/code/MOOS/trunk/Core/MOOSLIB/MOOSInstrument.cpp

## 3.8 CMOOSMsg Class Reference

`#include <MOOSMsg.h>`

### Public Member Functions

- **CMOOSMsg** ()
- **CMOOSMsg** (char cMsgType, const std::string &sKey, double dfVal, double dfTime=-1)
- **CMOOSMsg** (char cMsgType, const std::string &sKey, const std::string &sVal, double dfTime=-1)
- bool **IsDataType** (char cDataType)
- bool **IsDouble** ()
- bool **IsString** ()
- bool **IsSkewed** (double dfTimeNow, double *pdfSkew=NULL)
- bool **IsYoungerThan** (double dfAge)
- bool **IsType** (char cType)
- double **GetTime** ()
- double **GetDouble** ()
- std::string **GetString** ()
- std::string **GetKey** ()
- std::string **GetName** ()
- std::string **GetSource** ()
- std::string **GetCommunity** ()
- std::string **GetAsString** (int nFieldWidth=12)
- void **Trace** ()
- int **Serialize** (unsigned char *pBuffer, int nLen, bool bToStream=true)
- bool **operator<** (const **CMOOSMsg** &Msg) const

### Public Attributes

- char **m_cMsgType**
- char **m_cDataType**
- std::string **m_sKey**
- int **m_nID**
- double **m_dfTime**
- double **m_dfVal**
- double **m_dfVal2**
- std::string **m_sVal**
- std::string **m_sSrc**
- std::string **m_sOriginatingCommunity**

### 3.8.1 Detailed Description

MOOS Comms Messaging class. This is a class encapsulating the data which the MOOS Comms API shuttles between the MOOSDB and other clients

## 3.8.2 Constructor & Destructor Documentation

### 3.8.2.1 CMOOSMsg::CMOOSMsg ()

standard construction destruction

### 3.8.2.2 CMOOSMsg::CMOOSMsg (char *cMsgType*, const std::string & *sKey*, double *dfVal*, double *dfTime* = -1)

specialised construction

### 3.8.2.3 CMOOSMsg::CMOOSMsg (char *cMsgType*, const std::string & *sKey*, const std::string & *sVal*, double *dfTime* = -1)

specialised construction

## 3.8.3 Member Function Documentation

### 3.8.3.1 string CMOOSMsg::GetAsString (int *nFieldWidth* = 12)

format the message as string regardless of type

### 3.8.3.2 std::string CMOOSMsg::GetCommunity () [inline]

return the name of the MOOS community in which the orginator lives

### 3.8.3.3 double CMOOSMsg::GetDouble () [inline]

return double val of message

### 3.8.3.4 std::string CMOOSMsg::GetKey () [inline]

return the name of the message

### 3.8.3.5 std::string CMOOSMsg::GetSource () [inline]

return the name of the process (as registered with the DB) which posted this notification

### 3.8.3.6 std::string CMOOSMsg::GetString () [inline]

return string value of message

### 3.8.3.7 double CMOOSMsg::GetTime () [inline]

return time stamp of message

#### 3.8.3.8 bool CMOOSMsg::IsDataType (char *cDataType*)

check data type (MOOS_STRING or MOOS_DOUBLE)

#### 3.8.3.9 bool CMOOSMsg::IsDouble () [inline]

check data type is double

#### 3.8.3.10 bool CMOOSMsg::IsSkewed (double *dfTimeNow*, double * *pdfSkew* = NULL)

return true if mesage is substantially (SKEW_TOLERANCE) older than dfTimeNow if pdfSkew is not NULL, the time skew is returned in *pdfSkew

#### 3.8.3.11 bool CMOOSMsg::IsString () [inline]

check data type is string

#### 3.8.3.12 bool CMOOSMsg::IsType (char *cType*)

check message type MOOS_NOTIFY, REGISTER etc

#### 3.8.3.13 bool CMOOSMsg::IsYoungerThan (double *dfAge*)

return true if message is younger that dfAge

#### 3.8.3.14 void CMOOSMsg::Trace ()

print a summary of the message

### 3.8.4 Member Data Documentation

#### 3.8.4.1 char CMOOSMsg::m_cDataType

what kind of data is this? String,Double,Array?

#### 3.8.4.2 char CMOOSMsg::m_cMsgType

what type of message is this? Notification,Command,Register etc

#### 3.8.4.3 double CMOOSMsg::m_dfTime

double precision time stamp (UNIX time)

#### 3.8.4.4 int CMOOSMsg::m_nID

ID of message

#### 3.8.4.5 std::string CMOOSMsg::m_sKey

what is the variable name?

The documentation for this class was generated from the following files:

- /home/pnewman/code/MOOS/trunk/Core/MOOSLIB/MOOSMsg.h
- /home/pnewman/code/MOOS/trunk/Core/MOOSLIB/MOOSMsg.cpp

# Index


AddMOOSVariable
    CMOOSApp, 6

ClientLoop
    CMOOSCommClient, 13
Close
    CMOOSCommClient, 13
CMOOSApp, 5
    AddMOOSVariable, 6
    ConfigureComms, 7
    GetAppName, 7
    GetAppStartTime, 7
    GetIterateCount, 7
    GetLastIterateTime, 7
    GetMissionFileName, 7
    GetMOOSVar, 7
    GetTimeSinceIterate, 7
    IsSimulateMode, 7
    Iterate, 7
    m_bServerSet, 10
    m_bSimMode, 10
    m_bUseMOOSComms, 10
    m_Comms, 10
    m_dfAppStartTime, 10
    m_dfFreq, 10
    m_dfLastRunTime, 10
    m_lServerPort, 10
    m_MissionReader, 10
    m_MOOSVars, 10
    m_nCommsFreq, 10
    m_sAppName, 11
    m_sMissionFile, 11
    m_sServerHost, 11
    m_sServerPort, 11
    MOOSDebugWrite, 8
    OnConnectToServer, 8
    OnDisconnectFromServer, 8
    OnNewMail, 8
    OnStartUp, 8
    PublishFreshMOOSVariables, 8
    RegisterMOOSVariables, 8
    Run, 8
    SetAppFreq, 9
    SetCommsFreq, 9
    SetMOOSVar, 9
    SetServer, 9
    UpdateMOOSVariables, 9
    UseMOOSComms, 9
CMOOSCommClient, 12
CMOOSCommClient
    ClientLoop, 13
    Close, 13
    ConnectToServer, 14
    DoBanner, 14
    DoClientWork, 14
    FakeSource, 14
    Fetch, 14
    GetDescription, 14
    GetLocalIPAddress, 14
    HandShake, 14
    IsConnected, 14
    m_bConnected, 17
    m_bFakeSource, 17
    m_bMailPresent, 17
    m_bQuit, 17
    m_InBox, 17
    m_InLock, 17
    m_lPort, 17
    m_nFundamentalFreq, 18
    m_nInPendingLimit, 18
    m_nOutPendingLimit, 18
    m_OutBox, 18
    m_OutLock, 18
    m_pConnectCallBackParam, 18
    m_pDisconnectCallBackParam, 18
    m_pfnConnectCallBack, 18
    m_pfnDisconnectCallBack, 18
    m_pSocket, 19
    m_Registered, 19
    m_sDBHost, 19
    m_sMyName, 19
    Notify, 15
    OnCloseConnection, 15
    Peek, 15
    PeekMail, 15
    Post, 15
    Register, 15
    Run, 16
    ServerRequest, 16
    SetOnConnectCallBack, 16

SetOnDisconnectCallBack, 16
StartThreads, 16
THREAD_ID, 13
UnRegister, 17
CMOOSCommObject, 20
CMOOSCommPkt, 21
CMOOSCommPkt
m_bAllocated, 21
Serialize, 21
CMOOSCommServer, 23
CMOOSCommServer
DoBanner, 25
GetClientName, 25
GetMaxSocketFD, 25
HandShake, 25
IsUniqueName, 25
ListenLoop, 25
m_ClientSocketList, 27
m_pDisconnectCallBackParam, 27
m_pfnDisconnectCallBack, 27
m_pfnRxCallBack, 27
m_pFocusSocket, 27
m_pListenSocket, 28
m_pRxCallBackParam, 28
m_Socket2ClientMap, 28
m_SocketListLock, 28
OnNewClient, 25
PoisonClient, 26
ProcessClient, 26
Run, 26
ServerLoop, 26
SetOnDisconnectCallBack, 26
SetOnRxCallBack, 26
THREAD_ID, 25
TimerLoop, 27
CMOOSException, 29
CMOOSException, 29
CMOOSInstrument, 30
DoNMEACheckSum, 31
GetMagneticOffset, 31
InitialiseSensor, 31
InitialiseSensorN, 31
m_bPublishRaw, 31
m_dfMagneticOffset, 31
m_Port, 31
m_sResourceName, 32
Message2NMEA, 31
OnStartUp, 31
PublishRaw, 31
SetupPort, 31
CMOOSMsg, 33
CMOOSMsg, 34
GetAsString, 34
GetCommunity, 34
GetDouble, 34
GetKey, 34
GetSource, 34
GetString, 34
GetTime, 34
IsDataType, 34
IsDouble, 35
IsSkewed, 35
IsString, 35
IsType, 35
IsYoungerThan, 35
m_cDataType, 35
m_cMsgType, 35
m_dfTime, 35
m_nID, 35
m_sKey, 35
Trace, 35
ConfigureComms
CMOOSApp, 7
ConnectToServer
CMOOSCommClient, 14

DoBanner
CMOOSCommClient, 14
CMOOSCommServer, 25
DoClientWork
CMOOSCommClient, 14
DoNMEACheckSum
CMOOSInstrument, 31

FakeSource
CMOOSCommClient, 14
Fetch
CMOOSCommClient, 14

GetAppName
CMOOSApp, 7
GetAppStartTime
CMOOSApp, 7
GetAsString
CMOOSMsg, 34
GetClientName
CMOOSCommServer, 25
GetCommunity
CMOOSMsg, 34
GetDescription
CMOOSCommClient, 14
GetDouble
CMOOSMsg, 34
GetIterateCount
CMOOSApp, 7
GetKey
CMOOSMsg, 34
GetLastIterateTime

CMOOSApp, 7
GetLocalIPAddress
CMOOSCommClient, 14
GetMagneticOffset
CMOOSInstrument, 31
GetMaxSocketFD
CMOOSCommServer, 25
GetMissionFileName
CMOOSApp, 7
GetMOOSVar
CMOOSApp, 7
GetSource
CMOOSMsg, 34
GetString
CMOOSMsg, 34
GetTime
CMOOSMsg, 34
GetTimeSinceIterate
CMOOSApp, 7

HandShake
CMOOSCommClient, 14
CMOOSCommServer, 25

InitialiseSensor
CMOOSInstrument, 31
InitialiseSensorN
CMOOSInstrument, 31
IsConnected
CMOOSCommClient, 14
IsDataType
CMOOSMsg, 34
IsDouble
CMOOSMsg, 35
IsSimulateMode
CMOOSApp, 7
IsSkewed
CMOOSMsg, 35
IsString
CMOOSMsg, 35
IsType
CMOOSMsg, 35
IsUniqueName
CMOOSCommServer, 25
IsYoungerThan
CMOOSMsg, 35
Iterate
CMOOSApp, 7

ListenLoop
CMOOSCommServer, 25

m_bAllocated
CMOOSCommPkt, 21
m_bConnected
CMOOSCommClient, 17
m_bFakeSource
CMOOSCommClient, 17
m_bMailPresent
CMOOSCommClient, 17
m_bPublishRaw
CMOOSInstrument, 31
m_bQuit
CMOOSCommClient, 17
m_bServerSet
CMOOSApp, 10
m_bSimMode
CMOOSApp, 10
m_bUseMOOSComms
CMOOSApp, 10
m_cDataType
CMOOSMsg, 35
m_ClientSocketList
CMOOSCommServer, 27
m_cMsgType
CMOOSMsg, 35
m_Comms
CMOOSApp, 10
m_dfAppStartTime
CMOOSApp, 10
m_dfFreq
CMOOSApp, 10
m_dfLastRunTime
CMOOSApp, 10
m_dfMagneticOffset
CMOOSInstrument, 31
m_dfTime
CMOOSMsg, 35
m_InBox
CMOOSCommClient, 17
m_InLock
CMOOSCommClient, 17
m_lPort
CMOOSCommClient, 17
m_lServerPort
CMOOSApp, 10
m_MissionReader
CMOOSApp, 10
m_MOOSVars
CMOOSApp, 10
m_nCommsFreq
CMOOSApp, 10
m_nFundamentalFreq
CMOOSCommClient, 18
m_nID
CMOOSMsg, 35
m_nInPendingLimit
CMOOSCommClient, 18

m_nOutPendingLimit
CMOOSCommClient, 18
m_OutBox
CMOOSCommClient, 18
m_OutLock
CMOOSCommClient, 18
m_pConnectCallBackParam
CMOOSCommClient, 18
m_pDisconnectCallBackParam
CMOOSCommClient, 18
CMOOSCommServer, 27
m_pfnConnectCallBack
CMOOSCommClient, 18
m_pfnDisconnectCallBack
CMOOSCommClient, 18
CMOOSCommServer, 27
m_pfnRxCallBack
CMOOSCommServer, 27
m_pFocusSocket
CMOOSCommServer, 27
m_pListenSocket
CMOOSCommServer, 28
m_Port
CMOOSInstrument, 31
m_pRxCallBackParam
CMOOSCommServer, 28
m_pSocket
CMOOSCommClient, 19
m_Registered
CMOOSCommClient, 19
m_sAppName
CMOOSApp, 11
m_sDBHost
CMOOSCommClient, 19
m_sKey
CMOOSMsg, 35
m_sMissionFile
CMOOSApp, 11
m_sMyName
CMOOSCommClient, 19
m_Socket2ClientMap
CMOOSCommServer, 28
m_SocketListLock
CMOOSCommServer, 28
m_sResourceName
CMOOSInstrument, 32
m_sServerHost
CMOOSApp, 11
m_sServerPort
CMOOSApp, 11
Message2NMEA
CMOOSInstrument, 31
MOOSDebugWrite
CMOOSApp, 8
Notify
CMOOSCommClient, 15

OnCloseConnection
CMOOSCommClient, 15
OnConnectToServer
CMOOSApp, 8
OnDisconnectFromServer
CMOOSApp, 8
OnNewClient
CMOOSCommServer, 25
OnNewMail
CMOOSApp, 8
OnStartUp
CMOOSApp, 8
CMOOSInstrument, 31

Peek
CMOOSCommClient, 15
PeekMail
CMOOSCommClient, 15
PoisonClient
CMOOSCommServer, 26
Post
CMOOSCommClient, 15
ProcessClient
CMOOSCommServer, 26
PublishFreshMOOSVariables
CMOOSApp, 8
PublishRaw
CMOOSInstrument, 31

Register
CMOOSCommClient, 15
RegisterMOOSVariables
CMOOSApp, 8
Run
CMOOSApp, 8
CMOOSCommClient, 16
CMOOSCommServer, 26

Serialize
CMOOSCommPkt, 21
ServerLoop
CMOOSCommServer, 26
ServerRequest
CMOOSCommClient, 16
SetAppFreq
CMOOSApp, 9
SetCommsFreq
CMOOSApp, 9
SetMOOSVar
CMOOSApp, 9
SetOnConnectCallBack

CMOOSCommClient, 16
SetOnDisconnectCallBack
CMOOSCommClient, 16
CMOOSCommServer, 26
SetOnRxCallBack
CMOOSCommServer, 26
SetServer
CMOOSApp, 9
SetupPort
CMOOSInstrument, 31
StartThreads
CMOOSCommClient, 16

THREAD_ID
CMOOSCommClient, 13
CMOOSCommServer, 25
TimerLoop
CMOOSCommServer, 27
Trace
CMOOSMsg, 35

UnRegister
CMOOSCommClient, 17
UpdateMOOSVariables
CMOOSApp, 9
UseMOOSComms
CMOOSApp, 9