



Delta3D Version 2.4.0

# **dtNet::**

## **Reference Manual**



# Contents

---

<b>1</b>	<b>Main Page</b>	<b>1</b>
<b>2</b>	<b>Directory Documentation</b>	<b>3</b>
2.1	inc/dtNet/ Directory Reference . . . . .	3
2.2	src/dtNet/ Directory Reference . . . . .	4
2.3	inc/ Directory Reference . . . . .	5
2.4	src/ Directory Reference . . . . .	6
<b>3</b>	<b>Namespace Documentation</b>	<b>7</b>
3.1	dtNet Namespace Reference . . . . .	7
3.1.1	Detailed Description . . . . .	7
<b>4</b>	<b>Class Documentation</b>	<b>9</b>
4.1	ConnectionListener Class Reference . . . . .	9
4.1.1	Detailed Description . . . . .	9
4.1.2	Member Typedef Documentation . . . . .	10
4.1.2.1	sptr . . . . .	10
4.1.2.2	wptr . . . . .	10
4.1.3	Constructor & Destructor Documentation . . . . .	10
4.1.3.1	~ConnectionListener . . . . .	10
4.1.3.2	ConnectionListener . . . . .	10
4.1.4	Member Function Documentation . . . . .	10
4.1.4.1	create . . . . .	10
4.1.4.2	onConnect . . . . .	10
4.1.4.3	onConnectFailure . . . . .	10
4.1.4.4	onDisconnect . . . . .	10
4.1.4.5	onError . . . . .	10
4.1.4.6	onExit . . . . .	10
4.1.4.7	onFailure . . . . .	10
4.1.4.8	onNewConn . . . . .	10
4.1.4.9	onReceive . . . . .	10
4.2	ConnectionServer Class Reference . . . . .	11
4.2.1	Detailed Description . . . . .	11
4.2.2	Member Typedef Documentation . . . . .	11
4.2.2.1	sptr . . . . .	11
4.2.2.2	wptr . . . . .	11
4.2.3	Constructor & Destructor Documentation . . . . .	11
4.2.3.1	ConnectionServer . . . . .	11

4.2.3.2	~ConnectionServer . . . . .	11
4.2.4	Member Function Documentation . . . . .	11
4.2.4.1	create . . . . .	11
4.2.4.2	getNewConnectionParams . . . . .	12
4.2.4.3	onListenFailure . . . . .	12
4.2.4.4	onListenSuccess . . . . .	12
4.3	NetMgr Class Reference . . . . .	13
4.3.1	Detailed Description . . . . .	15
4.3.2	Member Typedef Documentation . . . . .	15
4.3.2.1	ConnectionIterator . . . . .	15
4.3.3	Constructor & Destructor Documentation . . . . .	15
4.3.3.1	NetMgr . . . . .	15
4.3.3.2	~NetMgr . . . . .	15
4.3.4	Member Function Documentation . . . . .	15
4.3.4.1	AddConnection . . . . .	15
4.3.4.2	GetIsServer . . . . .	15
4.3.4.3	GetNumConnections . . . . .	15
4.3.4.4	InitializeGame . . . . .	15
4.3.4.5	OnConnect . . . . .	15
4.3.4.6	OnConnectFailure . . . . .	16
4.3.4.7	OnDisconnect . . . . .	16
4.3.4.8	OnError . . . . .	16
4.3.4.9	OnExit . . . . .	16
4.3.4.10	OnFailure . . . . .	16
4.3.4.11	OnListenFailure . . . . .	16
4.3.4.12	OnListenSuccess . . . . .	16
4.3.4.13	OnNewConn . . . . .	16
4.3.4.14	OnReceive . . . . .	17
4.3.4.15	RemoveConnection . . . . .	17
4.3.4.16	SendPacket . . . . .	17
4.3.4.17	SetupClient . . . . .	17
4.3.4.18	SetupServer . . . . .	17
4.3.4.19	Shutdown . . . . .	17
4.3.5	Member Data Documentation . . . . .	17
4.3.5.1	mConnections . . . . .	17
4.3.5.2	mInitialized . . . . .	17
4.3.5.3	mIsServer . . . . .	18
4.3.5.4	mMutex . . . . .	18
<b>5</b>	<b>File Documentation</b>	<b>19</b>
5.1	connectionlistener.cpp File Reference . . . . .	19

---

5.2	connectionlistener.h File Reference . . . . .	20
5.3	connectionserver.cpp File Reference . . . . .	21
5.4	connectionserver.h File Reference . . . . .	22
5.5	dtnet.h File Reference . . . . .	23
5.6	export.h File Reference . . . . .	24
5.6.1	Define Documentation . . . . .	24
5.6.1.1	DT_NET_EXPORT . . . . .	24
5.7	mainpage.h File Reference . . . . .	25
5.7.1	Detailed Description . . . . .	25
5.8	netmgr.cpp File Reference . . . . .	26
5.9	netmgr.h File Reference . . . . .	27



## Main Page

---

Delta3D is an Open Source engine which can be used for games, simulations, or other graphical applications.

The Delta3D framework exists as a number of modules, each sitting in its own library, enclosed within its own namespace. At the very core lies the dtCore library. This contains basic, low-level functionality which is mostly required for all 3D applications written in C++.

Around and alongside this sit other supporting libraries, such as dtUtil (containing reusable features which are useful for most applications), dtTerrain (for rendering terrain databases), dtGame, **dtNet** (p. 7), etc.

Extensive online documentation is available from the Delta3D Docs section to help in using Delta3D.

The project's original reference guides generated by Doxygen from the source code may be viewed at the Delta3D API Documentation section.

To download source code, binaries, dependencies and sample datasets visit the Delta3D Downloads page.

For more about dependencies see the Delta3D Dependencies page.

The documentation you are looking at can be downloaded from [www.3draum.ch](http://www.3draum.ch).

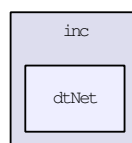
Enjoy!



# Directory Documentation

---

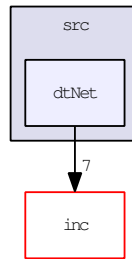
## 2.1 inc/dtNet/ Directory Reference



### Files

- file **connectionlistener.h**
- file **connectionserver.h**
- file **dtnet.h**
- file **export.h**
- file **mainpage.h**
- file **netmgr.h**

## 2.2 src/dtNet/ Directory Reference



### Files

- file **connectionlistener.cpp**
- file **connectionserver.cpp**
- file **netmgr.cpp**

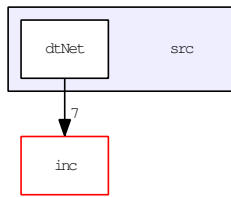
## 2.3 inc/ Directory Reference



### Directories

- directory **dtNet**

## 2.4 src/ Directory Reference



### Directories

- directory **dtNet**

# Namespace Documentation

---

### 3.1 dtNet Namespace Reference

The **dtNet** (p. 7) namespace contains networking classes.

#### Classes

- class **ConnectionListener**  
*Provides the interface to a GNE::Connection.*
- class **ConnectionServer**  
*This class is used as an interface to the GNE::Server connection.*
- class **NetMgr**  
*This class is used as the base of all networking applications.*

#### 3.1.1 Detailed Description

The **dtNet** (p. 7) namespace contains networking classes. **dtNet** (p. 7) is uses the Game Network Engine and HawkNL for backbone functionality.



# Class Documentation

---

## 4.1 ConnectionListener Class Reference

Provides the interface to a GNE::Connection.

```
#include <inc/dtNet/connectionlistener.h>
```

### Public Types

- typedef GNE::SmartPtr< **ConnectionListener** > **sptr**
- typedef GNE::WeakPtr< **ConnectionListener** > **wptr**

### Public Member Functions

- virtual ~**ConnectionListener** (void)
- virtual void **onConnect** (GNE::SyncConnection &conn)
- virtual void **onConnectFailure** (GNE::Connection &conn, const GNE::Error &error)
- virtual void **onDisconnect** (GNE::Connection &conn)
- virtual void **onError** (GNE::Connection &conn, const GNE::Error &error)
- virtual void **onExit** (GNE::Connection &conn)
- virtual void **onFailure** (GNE::Connection &conn, const GNE::Error &error)
- virtual void **onNewConn** (GNE::SyncConnection &conn)
- virtual void **onReceive** (GNE::Connection &conn)

### Static Public Member Functions

- static **sptr create** (**NetMgr** \*netMgr)  
*static method used to create a new **ConnectionListener** (p. 9)*

### Protected Member Functions

- **ConnectionListener** (**NetMgr** \*netMgr)

#### 4.1.1 Detailed Description

Provides the interface to a GNE::Connection. This class is used internally by the **NetMgr** (p. 13) and is typically not used directly by the end user. This class contains a reference to an instance of **NetMgr** (p. 13) and calls it's virtual methods, mimicking the GNE interface.

## 4.1.2 Member Typedef Documentation

4.1.2.1 `typedef GNE::SmartPtr<ConnectionListener> sptr`

4.1.2.2 `typedef GNE::WeakPtr<ConnectionListener> wptr`

## 4.1.3 Constructor & Destructor Documentation

4.1.3.1 `~ConnectionListener (void) [virtual]`

4.1.3.2 `ConnectionListener (NetMgr * netMgr) [protected]`

## 4.1.4 Member Function Documentation

4.1.4.1 `static sptr create (NetMgr * netMgr) [inline, static]`

static method used to create a new `ConnectionListener` (p. 9)

4.1.4.2 `void onConnect (GNE::SyncConnection & conn) [virtual]`

4.1.4.3 `void onConnectFailure (GNE::Connection & conn, const GNE::Error & error) [virtual]`

4.1.4.4 `void onDisconnect (GNE::Connection & conn) [virtual]`

4.1.4.5 `void onError (GNE::Connection & conn, const GNE::Error & error) [virtual]`

4.1.4.6 `void onExit (GNE::Connection & conn) [virtual]`

4.1.4.7 `void onFailure (GNE::Connection & conn, const GNE::Error & error) [virtual]`

4.1.4.8 `void onNewConn (GNE::SyncConnection & conn) [virtual]`

4.1.4.9 `void onReceive (GNE::Connection & conn) [virtual]`

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

- `connectionlistener.h`
- `connectionlistener.cpp`

## 4.2 ConnectionServer Class Reference

This class is used as an interface to the GNE::Server connection.

```
#include <inc/dtNet/connectionserver.h>
```

### Public Types

- typedef GNE::SmartPtr< **ConnectionServer** > **sptr**
- typedef GNE::WeakPtr< **ConnectionServer** > **wptr**

### Public Member Functions

- virtual ~**ConnectionServer** (void)
- virtual void **getNewConnectionParams** (GNE::ConnectionParams &params)
- virtual void **onListenFailure** (const GNE::Error &error, const GNE::Address &from, const GNE::ConnectionListener::sptr &listener)
- virtual void **onListenSuccess** (const GNE::ConnectionListener::sptr &listener)

### Static Public Member Functions

- static **sptr create** (int inRate, int outRate, **NetMgr** \*netMgr)  
*Method used to create a new instance of **ConnectionServer** (p. 11).*

### Protected Member Functions

- **ConnectionServer** (int inRate, int outRate, **NetMgr** \*netMgr)

#### 4.2.1 Detailed Description

This class is used as an interface to the GNE::Server connection. It is used internally by the **NetMgr** (p. 13) and is typically not used directly by the end user. This class takes in a reference to a **NetMgr** (p. 13) and calls its virtual methods to mimic the GNE::ServerConnectionListener's callbacks.

#### 4.2.2 Member Typedef Documentation

4.2.2.1 typedef GNE::SmartPtr<ConnectionServer> **sptr**

4.2.2.2 typedef GNE::WeakPtr<ConnectionServer> **wptr**

#### 4.2.3 Constructor & Destructor Documentation

4.2.3.1 **ConnectionServer** (int *inRate*, int *outRate*, **NetMgr** \* *netMgr*) [protected]

Parameters

**inRate** : the incoming bandwidth throttle

**outRate** : the outgoing bandwidth throttlw

**netMgr** : instance of a valid **NetMgr** (p. 13)

4.2.3.2 ~**ConnectionServer** (void) [virtual]

#### 4.2.4 Member Function Documentation

4.2.4.1 static **sptr create** (int *inRate*, int *outRate*, **NetMgr** \* *netMgr*) [inline, static]

Method used to create a new instance of **ConnectionServer** (p. 11).

4.2.4.2 void `getNewConnectionParams` (`GNE::ConnectionParams & params`) [virtual]

4.2.4.3 void `onListenFailure` (`const GNE::Error & error`, `const GNE::Address & from`, `const GNE::ConnectionListener::sptr & listener`) [virtual]

4.2.4.4 void `onListenSuccess` (`const GNE::ConnectionListener::sptr & listener`) [virtual]

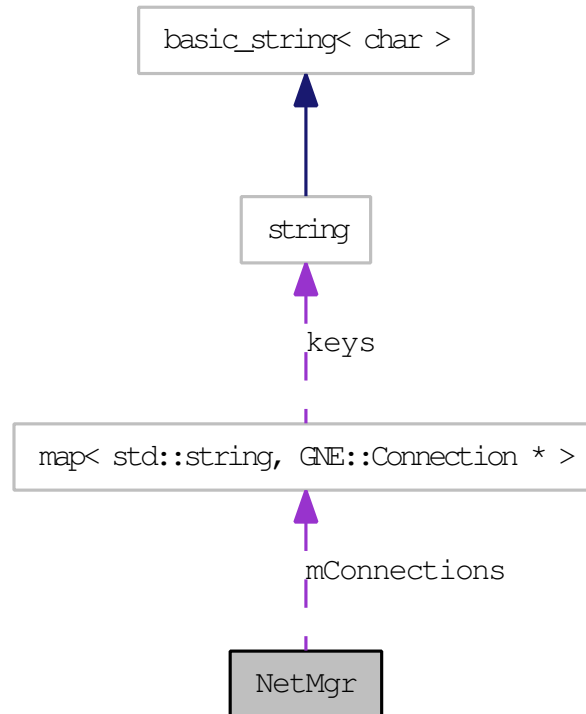
The documentation for this class was generated from the following files:

- `connectionserver.h`
- `connectionserver.cpp`

## 4.3 NetMgr Class Reference

This class is used as the base of all networking applications.

`#include <inc/dtNet/netmgr.h>` Collaboration diagram for NetMgr:



### Public Member Functions

- **NetMgr** ()
- bool **GetIsServer** () const  
*Is this instance setup as a server?*
- int **GetNumConnections** () const  
*Get the number of connections to the network.*
- void **InitializeGame** (const std::string &gameName, int gameVersion, const std::string &logFile)  
*Initialize the networking and game environment.*
- virtual void **OnConnect** (GNE::SyncConnection &conn)  
*called when the client is connected to the server*
- virtual void **OnConnectFailure** (GNE::Connection &conn, const GNE::Error &error)  
*A connection failed before or during the onConnect event.*
- virtual void **OnDisconnect** (GNE::Connection &conn)  
*The GNE::Connection has been disconnected.*
- virtual void **OnError** (GNE::Connection &conn, const GNE::Error &error)  
*A non-fatal error has occurred in the connection.*
- virtual void **OnExit** (GNE::Connection &conn)  
*called when the remote has gracefully closed the connection*

- virtual void **OnFailure** (GNE::Connection &conn, const GNE::Error &error)  
*A fatal error has occurred in the connection.*
- virtual void **OnListenFailure** (const GNE::Error &error, const GNE::Address &from, const GNE::ConnectionListener::sptr &listener)  
*callback to signal the connection to the socket failed*
- virtual void **OnListenSuccess** ()  
*virtual methods*
- virtual void **OnNewConn** (GNE::SyncConnection &conn)  
*called when the server receives a new connection*
- virtual void **OnReceive** (GNE::Connection &conn)  
*one or more GNE::Packets have been received*
- void **SendPacket** (const std::string &address, GNE::Packet &packet)  
*Send a packet to the given address.*
- bool **SetupClient** (const std::string &host, int portNum)  
*Setup and create a client to connect to the server.*
- bool **SetupServer** (int portNum)  
*Setup and create a server.*
- void **Shutdown** ()  
*Shutdown the networking.*

## Protected Types

- typedef std::map< std::string, GNE::Connection \* >::iterator **ConnectionIterator**

## Protected Member Functions

- virtual ~**NetMgr** ()
- void **AddConnection** (GNE::Connection \*connection)  
*Add a new GNE::Connection to the list of existing connections.*
- void **RemoveConnection** (GNE::Connection \*connection)  
*Remove an existing GNE::Connection from the list of connections.*

## Protected Attributes

- std::map< std::string, GNE::Connection \* > **mConnections**  
*A map of network address strings to Connections.*
- bool **mInitialized**  
*has the network been initialed yet?*
- bool **mIsServer**  
*are we a server?*
- GNE::Mutex **mMutex**

### 4.3.1 Detailed Description

This class is used as the base of all networking applications. It handles creating a server or a client and provides a convenient place to implement application-specific functionality.

It can be used as-is, but it's anticipated that the end-user will derive from this class and override the virtual methods as needed.

To use this class, create an instance and call **InitializeGame()** (p. 15); this will setup the internals of the network. Then call either **SetupServer()** (p. 17) or **SetupClient()** (p. 17) to start the networking. To end the networking, just call the **Shutdown()** (p. 17) method.

To pass data packets through the network, supply a GNE::Packet to **SendPacketToAll()**. A custom GNE::Packet must be registered with GNE after **InitilizeGame()** using:

```
GNE::PacketParser::defaultRegisterPacket<MyCustomPacket>();
```

### 4.3.2 Member Typedef Documentation

**4.3.2.1** `typedef std::map<std::string, GNE::Connection*>::iterator ConnectionIterator` [protected]

### 4.3.3 Constructor & Destructor Documentation

**4.3.3.1** `NetMgr ()`

**4.3.3.2** `~NetMgr ()` [protected, virtual]

### 4.3.4 Member Function Documentation

**4.3.4.1** `void AddConnection (GNE::Connection * connection)` [protected]

Add a new GNE::Connection to the list of existing connections. Internal method used to store the connection in a map.

Typically gets called from **OnConnect()** (p. 15) and **OnNewConn()** (p. 16) to save the connection for later use. Parameters

***connection*** : the connection to add to the list

**4.3.4.2** `bool GetIsServer () const` [inline]

Is this instance setup as a server?

**4.3.4.3** `int GetNumConnections () const` [inline]

Get the number of connections to the network.

**4.3.4.4** `void InitializeGame (const std::string & gameName, int gameVersion, const std::string & logFile)`

Initialize the networking and game environment. Initialize the network and setup the game parameters.

This method must be called before any other **NetMgr** (p. 13) methods. The supplied game name and game version are used to during the connection process to verify if the client/server match.

Parameters

***gameName*** : the name of the network game

***gameVersion*** : the version number of the game

***logFile*** : a filename to log networking debug information

**4.3.4.5** `void OnConnect (GNE::SyncConnection & conn)` [virtual]

called when the client is connected to the server Typically, this connection gets stored in a list for future reference.

Parameters

***conn*** : the new connection

See also **AddConnection()** (p. 15)

**4.3.4.6 void OnConnectFailure (GNE::Connection & *conn*, const GNE::Error & *error*) [virtual]**

A connection failed before or during the onConnect event. Parameters

***conn***,: The GNE::Connection that caused the failure

***error*** : The error describing the failure

**4.3.4.7 void OnDisconnect (GNE::Connection & *conn*) [virtual]**

The GNE::Connection has been disconnected. Parameters

***conn*** : the GNE::Connection that was just disconnected

**4.3.4.8 void OnError (GNE::Connection & *conn*, const GNE::Error & *error*) [virtual]**

A non-fatal error has occurred in the connection. Parameters

***conn***,: The GNE::Connection that caused the failure

***error*** : The error describing the failure

**4.3.4.9 void OnExit (GNE::Connection & *conn*) [virtual]**

called when the remote has gracefully closed the connection Parameters

***conn*** : the GNE::Connetion that just exited

**4.3.4.10 void OnFailure (GNE::Connection & *conn*, const GNE::Error & *error*) [virtual]**

A fatal error has occurred in the connection. Parameters

***conn***,: The GNE::Connection that caused the failure

***error*** : The error describing the failure

**4.3.4.11 void OnListenFailure (const GNE::Error & *error*, const GNE::Address & *from*, const GNE::ConnectionListener::sptr & *listener*) [virtual]**

callback to signal the connection to the socket failed Parameters

***error*** : The GNE:Error describing the failure

***from*** : The GNE::Address of the problem

***listener*** The GNE::ConnectionListen who triggered this failure

**4.3.4.12 void OnListenSuccess () [virtual]**

virtual methods callback to signal a connection is successful

**4.3.4.13 void OnNewConn (GNE::SyncConnection & *conn*) [virtual]**

called when the server receives a new connection Typically, this new connection gets stored in a list for future reference.

Parameters

***conn*** : the new connection

See also **AddConnection()** (p. 15)

**4.3.4.14 void OnReceive (GNE::Connection & *conn*) [virtual]**

one or more GNE::Packets have been received Parameters

: *conn* : the GNE::Connection which contains the GNE::Packets to be read

**4.3.4.15 void RemoveConnection (GNE::Connection \* *connection*) [protected]**

Remove an existing GNE::Connection from the list of connections. Internal method used to remove an existing connection from the list.

If the supplied connection is not in the list, it won't be removed. Parameters

***connection*** : the connection to remove from the list

**4.3.4.16 void SendPacket (const std::string & *address*, GNE::Packet & *packet*)**

Send a packet to the given address. Sends the supplied packet to all connections in the list.

If this is a server, it will send the packet to all existing connections. If this is a client, typically there will be only one connection: to the server.

Parameters

***address*** : the string representation of the address to send to or "all"

***packet*** : the GNE::Packet to send to the world

See also **AddConnection()** (p. 15)

**4.3.4.17 bool SetupClient (const std::string & *host*, int *portNum*)**

Setup and create a client to connect to the server. Create a client and try to connect to the supplied host name.

Parameters

***host*** : the name of the host to connect to

***portNum*** : the socket port number to use

Returns true if successful, false otherwise

**4.3.4.18 bool SetupServer (int *portNum*)**

Setup and create a server. Create and start the server network.

Parameters

***portNum*** : the socket port number to listen to

Returns true if successful, false otherwise

**4.3.4.19 void Shutdown ()**

Shutdown the networking. Perform a graceful shutdown of the network.

This will attempt to disconnect all currently active connections.

**4.3.5 Member Data Documentation****4.3.5.1 std::map<std::string, GNE::Connection\*> mConnections [protected]**

A map of network address strings to Connections.

**4.3.5.2 bool mInitialized [protected]**

has the network been initialed yet?

**4.3.5.3 bool mIsServer [protected]**

are we a server?

**4.3.5.4 GNE::Mutex mMutex [protected]**

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

- **netmgr.h**
- **netmgr.cpp**

## File Documentation

---

### 5.1 connectionlistener.cpp File Reference

```
#include <dtNet/connectionlistener.h>
```

```
#include <dtNet/netmgr.h>
```

```
#include <dtUtil/log.h>
```

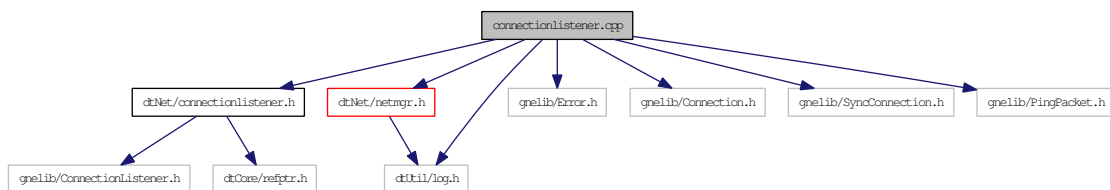
```
#include <gnelib/Error.h>
```

```
#include <gnelib/Connection.h>
```

```
#include <gnelib/SyncConnection.h>
```

```
#include <gnelib/PingPacket.h>
```

Include dependency graph for connectionlistener.cpp:

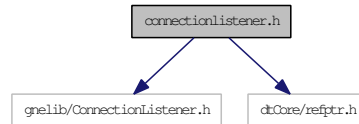


## 5.2 connectionlistener.h File Reference

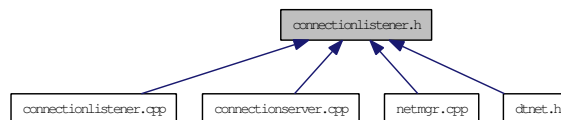
```
#include <gnelib/ConnectionListener.h>
```

```
#include <dtCore/refptr.h>
```

Include dependency graph for connectionlistener.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **ConnectionListener**  
*Provides the interface to a GNE::Connection.*

### Namespaces

- namespace **dtNet**  
*The **dtNet** (p. 7) namespace contains networking classes.*

## 5.3 connectionserver.cpp File Reference

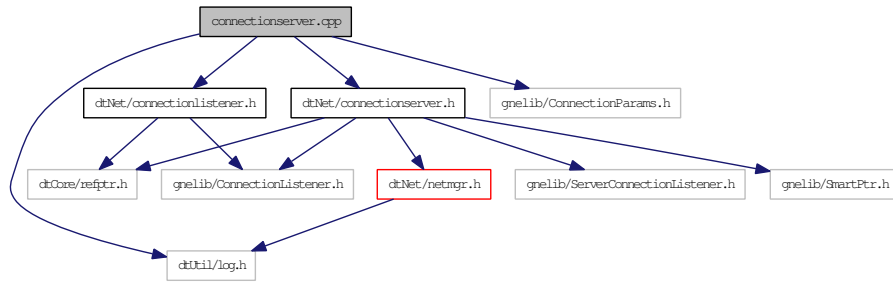
```
#include <dtNet/connectionserver.h>
```

```
#include <dtNet/connectionlistener.h>
```

```
#include <dtUtil/log.h>
```

```
#include <gnetlib/ConnectionParams.h>
```

Include dependency graph for connectionserver.cpp:



## 5.4 connectionserver.h File Reference

```
#include <dtNet/netmgr.h>
```

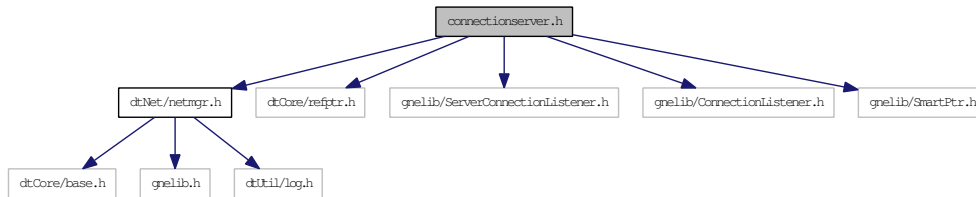
```
#include <dtCore/refptr.h>
```

```
#include <gnelib/ServerConnectionListener.h>
```

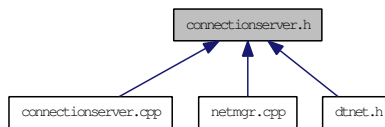
```
#include <gnelib/ConnectionListener.h>
```

```
#include <gnelib/SmartPtr.h>
```

Include dependency graph for connectionserver.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **ConnectionServer**

*This class is used as an interface to the GNE::Server connection.*

### Namespaces

- namespace **dtNet**

*The **dtNet** (p. 7) namespace contains networking classes.*

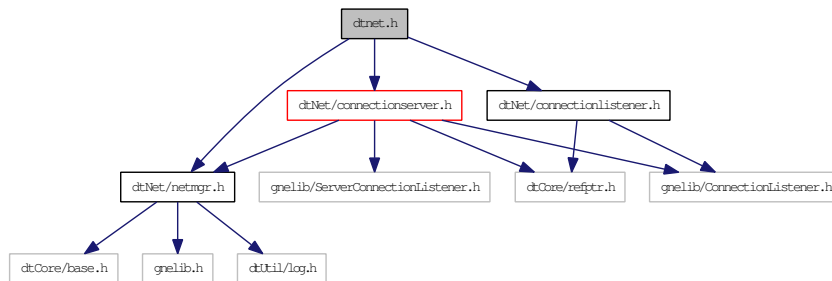
## 5.5 dtnet.h File Reference

```
#include <dtNet/netmgr.h>
```

```
#include <dtNet/connectionserver.h>
```

```
#include <dtNet/connectionlistener.h>
```

Include dependency graph for dtnet.h:



## Namespaces

- namespace **dtNet**

*The **dtNet** (p. 7) namespace contains networking classes.*

## 5.6 export.h File Reference

### Defines

- #define `DT_NET_EXPORT`

#### 5.6.1 Define Documentation

##### 5.6.1.1 #define `DT_NET_EXPORT`

## 5.7 mainpage.h File Reference

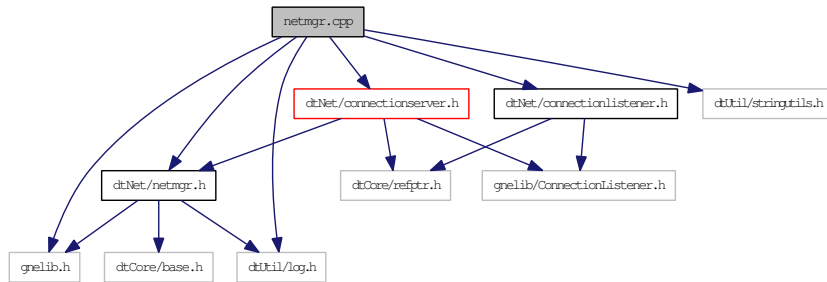
### 5.7.1 Detailed Description

This file contains Doxygen special commands and text for the **Main Page** (p. ??) and some other minor aspects of this documentation. It is not part of Delta3D.

## 5.8 netmgr.cpp File Reference

```
#include <dtNet/netmgr.h>
#include <dtNet/connectionlistener.h>
#include <dtNet/connectionserver.h>
#include <dtUtil/log.h>
#include <dtUtil/stringutils.h>
#include <gnelib.h>
```

Include dependency graph for netmgr.cpp:



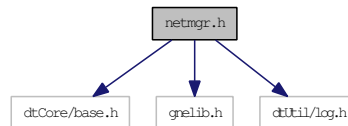
## 5.9 netmgr.h File Reference

```
#include <dtCore/base.h>
```

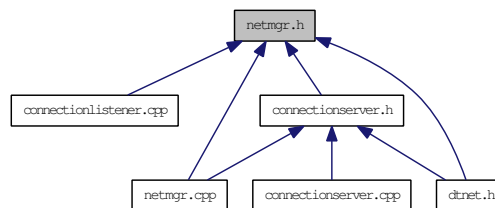
```
#include <gnetlib.h>
```

```
#include <dtUtil/log.h>
```

Include dependency graph for netmgr.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **NetMgr**

*This class is used as the base of all networking applications.*

### Namespaces

- namespace **dtNet**

*The **dtNet** (p. 7) namespace contains networking classes.*

# Index

---

## - Symbols -

~ConnectionListener  
dtNet::ConnectionListener, 10  
~ConnectionServer  
dtNet::ConnectionServer, 11  
~NetMgr  
dtNet::NetMgr, 15

## - A -

AddConnection  
dtNet::NetMgr, 15

## - C -

ConnectionIterator  
dtNet::NetMgr, 15  
ConnectionListener  
dtNet::ConnectionListener, 10  
connectionlistener.cpp, 19  
connectionlistener.h, 20  
ConnectionServer  
dtNet::ConnectionServer, 11  
connectionserver.cpp, 21  
connectionserver.h, 22  
create  
dtNet::ConnectionListener, 10  
dtNet::ConnectionServer, 11

## - D -

DT\_NET\_EXPORT  
export.h, 24  
dtNet, 7  
dtnet.h, 23  
dtNet::ConnectionListener, 9  
~ConnectionListener, 10  
ConnectionListener, 10  
create, 10  
onConnect, 10  
onConnectFailure, 10  
onDisconnect, 10  
onError, 10  
onExit, 10  
onFailure, 10  
onNewConn, 10  
onReceive, 10  
sptr, 10  
wptr, 10  
dtNet::ConnectionServer, 11  
~ConnectionServer, 11  
ConnectionServer, 11  
create, 11  
getNewConnectionParams, 11  
onListenFailure, 12  
onListenSuccess, 12  
sptr, 11  
wptr, 11  
dtNet::NetMgr, 13  
~NetMgr, 15  
AddConnection, 15  
ConnectionIterator, 15  
GetIsServer, 15

GetNumConnections, 15  
InitializeGame, 15  
mConnections, 17  
mInitialized, 17  
mIsServer, 17  
mMutex, 18  
NetMgr, 15  
OnConnect, 15  
OnConnectFailure, 15  
OnDisconnect, 16  
OnError, 16  
OnExit, 16  
OnFailure, 16  
OnListenFailure, 16  
OnListenSuccess, 16  
OnNewConn, 16  
OnReceive, 16  
RemoveConnection, 17  
SendPacket, 17  
SetupClient, 17  
SetupServer, 17  
Shutdown, 17

## - E -

export.h, 24  
DT\_NET\_EXPORT, 24

## - G -

GetIsServer  
dtNet::NetMgr, 15  
getNewConnectionParams  
dtNet::ConnectionServer, 11  
GetNumConnections  
dtNet::NetMgr, 15

## - I -

inc/ Directory Reference, 5  
inc/dtNet/ Directory Reference, 3  
InitializeGame  
dtNet::NetMgr, 15

## - M -

mainpage.h, 25  
mConnections  
dtNet::NetMgr, 17  
mInitialized  
dtNet::NetMgr, 17  
mIsServer  
dtNet::NetMgr, 17  
mMutex  
dtNet::NetMgr, 18

## - N -

NetMgr  
dtNet::NetMgr, 15  
netmgr.cpp, 26  
netmgr.h, 27

## - O -

OnConnect

dtNet::NetMgr, 15  
onConnect  
dtNet::ConnectionListener, 10  
OnConnectFailure  
dtNet::NetMgr, 15  
onConnectFailure  
dtNet::ConnectionListener, 10  
OnDisconnect  
dtNet::NetMgr, 16  
onDisconnect  
dtNet::ConnectionListener, 10  
OnError  
dtNet::NetMgr, 16  
onError  
dtNet::ConnectionListener, 10  
OnExit  
dtNet::NetMgr, 16  
onExit  
dtNet::ConnectionListener, 10  
OnFailure  
dtNet::NetMgr, 16  
onFailure  
dtNet::ConnectionListener, 10  
OnListenFailure  
dtNet::NetMgr, 16  
onListenFailure  
dtNet::ConnectionServer, 12  
OnListenSuccess  
dtNet::NetMgr, 16  
onListenSuccess  
dtNet::ConnectionServer, 12  
OnNewConn  
dtNet::NetMgr, 16  
onNewConn  
dtNet::ConnectionListener, 10  
OnReceive  
dtNet::NetMgr, 16  
onReceive  
dtNet::ConnectionListener, 10

**- R -**

RemoveConnection  
dtNet::NetMgr, 17

**- S -**

SendPacket  
dtNet::NetMgr, 17  
SetupClient  
dtNet::NetMgr, 17  
SetupServer  
dtNet::NetMgr, 17  
Shutdown  
dtNet::NetMgr, 17  
sptr  
dtNet::ConnectionListener, 10  
dtNet::ConnectionServer, 11  
src/ Directory Reference, 6  
src/dtNet/ Directory Reference, 4

**- W -**

wptr  
dtNet::ConnectionListener, 10  
dtNet::ConnectionServer, 11