# Naval Battle Client/Server Protocol

*Protocol v0.1*

Author:

Julien Amacher

Contenu

[Naval Battle Client/Server Protocol 1](#_Toc385324429)

[Messages format 3](#_Toc385324430)

[Transport protocol 3](#_Toc385324431)

[Timeout 3](#_Toc385324432)

[Beaconing 3](#_Toc385324433)

[GetGameParameters 4](#_Toc385324434)

[GetGameParameters request : 4](#_Toc385324435)

[GetGameParameters response : 4](#_Toc385324436)

[Connect 4](#_Toc385324437)

[With password : 4](#_Toc385324438)

[Without supplying a password : 5](#_Toc385324439)

[Connect response : 5](#_Toc385324440)

[Disconnect 6](#_Toc385324441)

[Disconnect request : 6](#_Toc385324442)

[Disconnect response : 6](#_Toc385324443)

[Play 6](#_Toc385324444)

[AttackCoordinate request : 7](#_Toc385324445)

[AttackCoordinate response : 7](#_Toc385324446)

[AttackedCoordinate request : 9](#_Toc385324447)

[EndOfGame : 10](#_Toc385324448)

[UserDisconnect : 10](#_Toc385324449)

[PositionMyBoats : 11](#_Toc385324450)

[PositionMyBoats request : 11](#_Toc385324451)

[PositionMyBoats response : 11](#_Toc385324452)

[GameStart : 12](#_Toc385324453)

[Nop (“NO OPeration”) : 12](#_Toc385324454)

[ChatSend message 13](#_Toc385324455)

[ChatSend request : 13](#_Toc385324456)

[ChatSend response : 13](#_Toc385324457)

[ChatReceive message 13](#_Toc385324458)

## Messages format

Requests and responses are XML 1.0 encoded in UTF-8. Both request and response are terminated with the sequence *\r\n\r\n*

Every request and response makes use of the “protocol” attribute which denotes the protocol’s version to use. Servers always answer with the same protocol the request used.

For now, since there is only one protocol, this attribute must be set to **1.0**

<request type=”**…**” protocol=”**1.0**”>

<response type=”**…**” protocol=”**1.0**”>

## Transport protocol

Servers listen on TCP port 36817 for incoming connections in ipv4 mode.

## Timeout

Servers and clients should close the connection when no data is received from the other side within a determined time frame: t=30s. Clients must make use of the **nop** message in order to keep the connection open. Consider sending a **nop** message when you have not sent a message in a 10 second window.

## Beaconing

Servers send, at fixed intervals of 10 seconds, an ipv4 UDP broadcast beacon on port 36818 used to advertise itself on the network, which contents is as follows:

<?xml version=”1.0” ? standalone=”yes”>

<response type="**endpointsDiscovery**" protocol=”1.0”>

<name>**SuperServer**</name>

<auth>**true**</auth>

<players>**2**</players>

<is\_full>**true**</is\_full>

<size>M</size>

</response>

auth: **true** if users should provide a shared secret (“password”) when connecting, **false** if no shared secret is required.

size : S = small, M = medium, L = large

Clients actively listening to these messages will then know the source address of this message is an endpoint itself (there are no so called “relay nodes”)

## GetGameParameters

This request is used to gather information on the current game. It is mainly used when a client wants to connect to a server but doesn’t have any information on it because it is not on the same network and, thus, does not receive its beacon.

### GetGameParameters request :

<?xml version=”1.0” standalone=”yes” ?>

<request type=”**getGameParameters**” protocol=”1.0” />

### GetGameParameters response :

The response content is the same as in the beaconing message sent by servers.

<?xml version=”1.0” standalone=”yes” ?>

<response type=”**getGameParameters**” protocol=”1.0”>

<name>**SuperServer**</name>

<auth>**true**</auth>

<players>**2**</players>

<full>**true**</full>

<size>**M**</size>

</response>pla

## Connect

First message to send to the server; used to connect or authenticate yourself.

### With password :

<?xml version=”1.0” standalone=”yes” ?>

<request type=”**connect**” protocol=”1.0”>

<username>**Luc**</username>

<digest type=”**shared\_secret**”>

<password>**sosecure**</password>

</digest>

</request>

|  |  |
| --- | --- |
| Field | Description |
| username | The client’s username |
| digest[type] | Fixed: only **shared\_secret** is supported |
| digest > password | The shared secret |

### Without supplying a password :

<?xml version=”1.0” standalone=”yes” ?>

<request type=”**connect**” protocol=”1.0”>

<username>**Luc**</username>

<digest type=”**none**”/>

</digest>

</request>

### Connect response :

<?xml version=”1.0” standalone=”yes” ?>

<response type=”**connect**” protocol=”1.0”>

<result>**authenticated**</result>

</response>

|  |  |
| --- | --- |
| Field | Description |
| result | **authenticated :** issued when the shared secret is accepted  **connected :** issued when a connection attempt is successful  **authentication\_error** : issued when an authentication attempt is not successful  **connection\_error** : issued when a connection attempt is not successful |

In case something went wrong :

<?xml version=”1.0” standalone=”yes” ?>

<response type=”**connect**” protocol=”1.0”>

<result>**authentication\_error**</result>

<error>

<code>**1**</code>

<description>**The provided username is already connected to the server.**</description>

</error>

</response>

Error codes :

|  |  |
| --- | --- |
| Code | Meaning |
| 1 | The provided username is already connected to the server |
| 2 | The provided shared secret is incorrect (“bad password”) |
| 3 | Server is full |
| 4 | Your endpoint is not allowed to connect |
| 5 | Authentication is not required but you authenticated anyway |

Please note: connection to a server which has authentication disabled will always result in **authentication\_error** in case a shared secret was supplied in the connection request.

## Disconnect

This message is sent by clients who wish to disconnect.

### Disconnect request :

<?xml version=”1.0” standalone=”yes” ?>

<request type=”**disconnect**” protocol=”1.0” />

### Disconnect response :

No response.

## Play

This request is used to attack a coordinate on the opponent’s map. Only applicable when it’s your turn to play and the game has not finished and has begun. Coordinates are measured from the top left (x is horizontal and y is vertical ; x >= 0 and y >= 0) : (x=0, y=0) is the top left coordinate. A play request cannot be sent before a play response has been received. Thus, this request is entirely synchronous.

### AttackCoordinate request :

<?xml version=”1.0” standalone=”yes” ?>

<request type=”**attackCoordinate**” protocol=”1.0”>

<coordinate>

<x>**3**</x>

<y>**4**</y>

<coordinate>

</request>

|  |  |
| --- | --- |
| Field | Description |
| coordinate > x | Horizontal point to attack |
| coordinate > y | Vertical point to attack |

### AttackCoordinate response :

In this example, a mine has been hit and ships around it were damaged and sinked : note that multiple ships can be damaged by a single mine!

<?xml version=”1.0” standalone=”yes” ?>

<response type=”**attackCoordinate**” protocol=”1.0”>

<result>**success**</result>

<coordinates>

<coordinate x=”**5**” y=”**5**” type=”**mine**” /> **// You hit a mine on (5,5) (where you played)**

<coordinate x=”**5**” y=”**6**” type=”**damaged**” /> **// The mine damaged a ship on (5,6)**

<coordinate x=”**6**” y=”**6**” type=”**sinked**” /> **// The mine finished damaging a ship on (6,6)**

<coordinates>

</response>

Another example :

<?xml version=”1.0” standalone=”yes” ?>

<response type=”**attackCoordinate**” protocol=”1.0”>

<result>**success**</result>

<coordinates>

<coordinate x=”**10**” y=”**16**” type=”**nothing**” /> **// You hit nothing on (10,16)**

<coordinates>

</response>

Another example :

<?xml version=”1.0” standalone=”yes” ?>

<response type=”**attackCoordinate**” protocol=”1.0”>

<result>**success**</result>

<coordinates>

<coordinate x=”**0**” y=”**0**” type=”**damaged**” /> **// You hit (but not sank) a ship on (0,0)**

<coordinates>

</response>

Another example :

<?xml version=”1.0” standalone=”yes” ?>

<response type=”**attackCoordinate**” protocol=”1.0”>

<result>**success**</result>

<coordinates>

<coordinate x=”**5**” y=”**5**” type=”**satellite**”/> **// You hit a Satellite bonus on (5,5)**

<coordinate x=”**10**” y=”**9**” type=”**revealed**” **has\_ship**=”**true**” /> **// The bonus revealed (10,9) and there’s a ship on that position**

<coordinate x=”**7**” y=”**7**” type=”**revealed**” **has\_ship**=”**true**” /> **// The bonus revealed (7,7) and there’s a ship on that position**

<coordinate x=”**1**” y=”**1**” type=”**revealed**” **has\_ship**=”**false**” /> **// The bonus also revealed (1,1) but there’s nothing there**

<coordinates>

</response>

|  |  |
| --- | --- |
| Field | Description |
| coordinates > coordinate[type] | **damaged** : your fire has damaged part of a ship. The coordinate indicates the position of the damage.  **sinked** : your fire has damaged and sinked a ship. The coordinate indicates the position of the last part of the ship that was still standing.  **mine**: your fire trigged an underwater mine.  **nothing** :nothing has happened, you fired in the sea.  **satellite :** your fire got you the Satellite bonus and some coordinates have been revealed to you. The attribute **has\_ship** is **true** in case a boat is present at that location on your opponent’s map. |

In case something went wrong :

<?xml version=”1.0” standalone=”yes” ?>

<response type=”**attackCoordinate**” protocol=”1.0”>

<result>**error**</result>

<error>

<code>**1**</code>

<description>**You already fired on this position.**</description>

</error>

</response>

|  |  |
| --- | --- |
| Code | Meaning |
| 1 | You already fired on this position (or a mine did) |
| 2 | Your coordinates were invalid (off grid or empty) |
| 3 | It’s not your turn to play |
| 4 | Game ended or has not yet started |

### AttackedCoordinate request :

**WARNING**: this message is a **request** (because it is purely asynchronous)

<?xml version=”1.0” standalone=”yes” ?>

<request type=”**attackedCoordinate**” protocol=”1.0”>

<username>**Luc**</username>

<coordinates>

<coordinate x=”**5**” y=”**5**” type=”**mine**” /> **// Your opponent has hit a mine on (5,5)**

<coordinate x=”**5**” y=”**6**” type=”**damaged**” /> **// The mine damaged your ship on (5,6)**

<coordinate x=”**6**” y=”**6**” type=”**sinked**” /> **// The mine finished damaging your ship on (6,6)**

</coordinates>

</request>

|  |  |
| --- | --- |
| Field | Description |
| username | Username of the attacker |
| coordinates > coordinate[type] | **damaged** : the opponent’s fire has damaged part of your ship. The coordinate indicates the position of the damage.  **sinked** : the opponent’s fire has damaged and sinked your ship. The coordinate indicates the position of the last part of your ship that was still standing.  **mine**: the opponent’s fire trigged an underwater mine  **nothing** :nothing has happened, the opponent’s fired in the sea |

## **EndOfGame** :

This message is received when the game ends.

**WARNING**: this message is a **request** (because it is purely asynchronous)

<?xml version=”1.0” standalone=”yes” ?>

<request type=”**endOfGame**” protocol=”1.0”>

<type>**winner**</type>

<winner>**Paul**</winner>

</request>

|  |  |
| --- | --- |
| Field | Description |
| type | **winner** : in case there is a winner  **cannot\_continue\_game** : game ended without a winner. For example, your opponent disconnected. |
| winner | The winner’s username. Is meaningless if type is not “winner” |

## **UserDisconnect** :

This message is received when the opponent disconnects (either because he chose to or because a timeout was reached). An **endOfGame** message should follow.

**WARNING**: this message is a **request** (because it is purely asynchronous)

<?xml version=”1.0” standalone=”yes” ?>

<request type=”**userDisconnect**” protocol=”1.0”>

<username>**Paul**</username>

</request>

|  |  |
| --- | --- |
| Field | Description |
| username | Username of the player who disconnected |

## **PositionMyBoats** :

Used to tell the server where you placed your boats; the game cannot begin unless both parties send this message.

### PositionMyBoats request :

<?xml version=”1.0” standalone=”yes” ?>

<request type=”**positionMyBoats**” protocol=”1.0”>

<boats>

<boat x=”**5**” y=”**7**” size=”**5**” orientation=”**vertical**” />

<boat x=”**10**” y=”**10**” size=”**5**” orientation=”**vertical**” />

<boat x=”**10**” y=”**0**” size=”**5**” orientation=”**horizontal**” />

</boats>

</request>

|  |  |
| --- | --- |
| Field | Description |
| boats > boat[x] | The horizontal position of the boat |
| boats > boat[y] | The vertical position of the boat |
| boats > boat[size] | The size of the boat |
| boats > boat[orientation] | The orientation of the boat. Can be either **horizontal** or **vertical** |

### PositionMyBoats response :

<?xml version=”1.0” standalone=”yes” ?>

<response type=”**positionMyBoats**” protocol=”1.0”>

<result>**success**</result>

</response>

|  |  |
| --- | --- |
| Field | Description |
| result | **success** or **error**  In case of **success**, wait for the **gameStart** message. In case of **error**, modify your request and submit again. |

In case something went wrong :

<?xml version=”1.0” standalone=”yes” ?>

<response type=”**positionMyBoats**” protocol=”1.0”>

<result>**error**</result>

<error>

<code>**2**</code>

<description>**Your boats are overlapping.**</description>

</error>

</response>

|  |  |
| --- | --- |
| Code | Meaning |
| 1 | Your coordinates were invalid (off grid or empty) |
| 2 | Your boats are overlapping |
| 3 | Your boats are already positioned |

## GameStart :

This message is received when both players have positioned their ships and are ready to start the game. Only the server can send this message.

**WARNING**: this message is a **request** (because it is purely asynchronous)

<?xml version=”1.0” standalone=”yes” ?>

<request type=”**gameStart**” protocol=”1.0”>

<first\_to\_play>**Luc**</first\_to\_play>

</request>

|  |  |
| --- | --- |
| Field | Description |
| first\_to\_play | Username of the player who will fire the first round. This is randomly determined by the server. If it’s not your username, wait for the opponent to fire (**attackedPosition** message) |

## Nop (“NO OPeration”) :

This message is used to send a periodic heartbeat. Can be sent by any client and by the server.

**WARNING** : when this message is sent by the server, the message type is **request**, NOT **response**. No response has to be provided to such message.

<?xml version=”1.0” standalone=”yes” ?>

<request type=”**nop**” protocol=”1.0” />

## **ChatSend message**

When you want to speak to your opponent.

### **ChatSend** request :

<?xml version=”1.0” standalone=”yes” ?>

<request type=”**chatSend**” protocol=”1.0”>

<nonce>**DZVFQEN5**</nonce>

<text>**Ohhay how u doing?**</text>

</request>

|  |  |
| --- | --- |
| Field | Description |
| nonce | A random value (optional) |
| text | What you want to say!  Maximum 200 chars.  **WARNING**  Cannot contain two consecutive **\r\n** |

### **ChatSend** response :

The server responds with the same nonce provided in the request, confirming it has been delivered.

<?xml version=”1.0” standalone=”yes” ?>

<response type=”**chatSend**” protocol=”1.0”>

<nonce>**DZVFQEN5**</nonce>

</response>

## **ChatReceive** message

Received when your opponent says something.

**WARNING**: this message is a **request** (because it is purely asynchronous)

<?xml version=”1.0” standalone=”yes” ?>

<request type=”**chatReceive**” protocol=”1.0”>

<username>**Luc**</username>

<text>**I like potatoes**</text>

</request>

|  |  |
| --- | --- |
| Field | Description |
| username | Username of the user speaking |
| text | What he/she has to say |