Requests and Responses for Battleship

The request/response patterns for the Battleship server. You send your request using a URL that follows the format [base URL]/[request type]. For example, if I wanted a game-state update, I would send the request to [base URL]/Update, where [base URL] is replaced with the URL below. You'll note that the request type is listed in parentheses for each request section below. The server reads URLs as case sensitive, so follow exact spelling and casing to avoid errors.

If a request is invalid (invalid syntax, invalid data, unknown request, incomplete request, incorrect game id, incorrect player id, invalid command based on game state, etc), you will get an error message back (formatted as XML, of course).

Upon making your first request create or join a game, the server will send you back a cookie. You must "remember" the cookie it sends you and send it with every subsequent request. If you do not, the server will not know who is making the request and you will be unable to continue your game. You must remember that your cookie is game specific. If you've just played a game and you wish to play another, you must first clear your cookie before making the new game quest. This way, you'll receive the correct cookie for the new game.

Base URL: http://dickerson.neumont.edu:8080/Battleship/GameRequest/

Valid Ship Types are:

Ship Name	Size
Carrier	5
Battleship	4
Submarine	3
Cruiser	3
PatrolBoat	2

Valid Ship Directions are: DOWN, UP, LEFT, RIGHT
AI Robots are (in order of difficulty, smartest first): Edison, Geeves, Robby
Valid game states are: WaitingFor2nd, WaitingForShips, InProgress, Finished, Forfeited, TimedOut
Game board is set up so that the letters are on the top and the numbers are along the side like this:

New Game:

Request (NewGame)

To wait for another person to join your game

```
<request>
        <playerID>King</playerID>
</request>
```

-or-

To play against Edison

```
<request>
        <playerID>King</playerID>
        <robot>Edison</robot>
</request>
```

Response: `<response><gameID>1679618673</gameID></response>`

Game List:

Request (GameList):

```
<request></request>
```

Response:

```
<response>
        <game>
                <gameID>1679618673</gameID>
                <turn>King</turn>
                <state>WaitingFor2nd</state>
        </game>
        <game>
                <gameID>1679618674</gameID>
                <turn>Friend</turn>
                <state>WaitingFor2nd</state>
        </game>
</response>
```

Join Game:

Request (Join):

```
<request>
        <playerID>Aaron</playerID>
        <gameID>1679618673</gameID>
```

```
</request>
```

Response:

```
<response>
	<gameID>1679618673</gameID>
	<result>Aaron Successfully Joined</result>
</response>
```

Place Ship:

Request (PlaceShip):

```
<request>
	<coordinates>B1</coordinates>
	<direction>DOWN</direction>
	<ship>Carrier</ship>
</request>
```

Response:

```
<response>
	<gameID>1679618673</gameID>
	<result>Successfully placed ship Aircraft Carrier</result>
</response>
```

Requesting Game State:

Request (Update):

```
<request></request>
```

Response:

Before anyone has actually fired

```
<response>
	<gameID>1679618673</gameID>
	<game>
		<gameID>1679618673</gameID>
		<turn>King</turn>
		<state>InProgress</state>
	</game>
</response>
```

-or-

If the result is a hit or a miss (not sunk)

```
<response>
	<gameID>1679618673</gameID>
	<game>
		<gameID>1679618673</gameID>
		<turn>Aaron</turn>
		<state>InProgress</state>
		<last>
			<coordinate>B2</coordinate>
			<status>Hit</status>
			<playerID>King</playerID>
		</last>
	</game>
</response>
```

-or-

If the result is a hit, sunk a ship, and the game is now over

```
<response>
	<gameID>1679618673</gameID>
	<game>
		<gameID>1679618673</gameID>
		<turn>King</turn>
		<state>Finished</state>
		<last>
			<coordinate>G10</coordinate>
			<status>Sunk</status>
			<playerID>Aaron</playerID>
			<ship>Carrier</ship>
		</last>
		<winner>Aaron</winner>
	</game>
</response>
```

Firing:

Request (Fire):

```
<request>
	<coordinates>F8</coordinates>
</request>
```

Response:

If you missed

```
<response>
	<gameID>1679618673</gameID>
	<result>Miss</result>
</response>
```

- or -

If you hit

```
<response>
	<gameID>1679618673</gameID>
	<result>Hit</result>
</response>
```

- or -

If you sunk a ship

```
<response>
	<gameID>1679618673</gameID>
	<result>Sunk</result>
	<ship>Cruiser</ship>
</response>
```

Forfeit:

Request (Forfeit)

```
<request></request>
```

Response:

```
<response>
	<gameID>815307830</gameID>
	<result>Player Aaron has forfeited.</result>
</response>
```

Errors:

If you send a request that the server doesn't understand, you will get this response:

```
<response>
	<error>I'm sorry, I didn't understand your request.</error>
</response>
```

If you send a request that wasn't valid XML, you will get this response:

```
<response>
	<error>Your request was not valid XML</error>
</response>
```

Other errors vary, but they always follow the format:

```
<response><error>[[error text]]</error></response>
```