Scrabble Protocol

v2.0.0

Contents

Changelog

- v1.0.0 *(16-01-2022)*
  - Initial protocol version
- v1.1.0 *(26-01-2022)*
  - Add `E010` : request game for wrong number of players
  - Add `E011` : word not connected to other tiles
  - Define `TEAMPLAY` flag for server
- v1.2.0 *(31-01-2022)*
  - Add `E012` : Client has already announced themselves
  - Add `E013` : Client has not yet announced themselves
  - Add `E014` : Client is not in a game
  - Add `E015` : Game already started
- v2.0.0 *(31-01-2022)*
  - Adapt happy flow to include `NEWTILES` after `STARTGAME`

Introduction

In this document, I will try to explain the main rules of this protocol. This will be supported by several UML-diagrams. The protocol is made to play a game of Scrabble on a server, where 2 to 4 clients can participate as players.
Just like every project, this document and/or protocol will contain mistakes. If you find any, please open a GitLab issue here.

Please note: This protocol is subject to change, but all major updates will be announced in the Protocol D group in Microsoft Teams. Keep an eye on this channel to stay up-to-date with the changes.

Definitions

The keywords “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119.

Delimiters

Each message, either from client to server or from server to client, ends with a message separator. This message separator is defined in the `Protocol` class as a static final character. This character is called the `MESSAGE_SEPARATOR` within the class, and is an ASCII record separator (representation, decimal, hex) as ($\boxed{\text{RS}}$, 30, 0x1E).
The command and its arguments (and the arguments itself) are seperated by a unit separator. This character is called the `UNIT_SEPARATOR` in the `Protocol` class, and is an ASCII unit separator ($\boxed{\text{US}}$, 31, 0x1F).

An example message would thus look as follows: `COMMAND`$\boxed{\text{US}}$`ARGUMENT1`$\boxed{\text{US}}$`ARGUMENT2`$\boxed{\text{RS}}$

Coordinates

The coordinates on the Scrabble board will be described by a letter for each column and a number for each row. The top left corner is thus coordinate `A1`, the top right corner `O1`, the bottom left corner `A15` and the bottom right corner `O15`.

Tiles

All tiles are by default capitalized. This also means that when making a move, the word should be communicated in all-caps. Blank tiles are represented by an exclamation mark `!`, until they are placed on the board. From that point onwards, they will be represented by a lowercase letter. If one want to make a move using a blank tile, they will have to communicate the word in uppercase, except the letter which uses a blank tile, which will communicated in lowercase.

Commands

The protocol supports the following basic commands:

```
ANNOUNCE
WELCOME
REQUESTGAME
INFORMQUEUE
STARTGAME
NOTIFYTURN
MAKEMOVE
NEWTILES
INFORMMOVE
ERROR
GAMEOVER
PLAYERDISCONNECTED
```

Furthermore, it also supports some chat commands:

```
SENDCHAT
NOTIFYCHAT
```

Other commands for extra features can be requested later in the project.

ANNOUNCE

The client announces itself to the server, together with its name and the flags of the supported extra features. As of now, the only supported flag by the protocol is `CHAT`. If no flags are specified (there is no second argument), the server will assume that no extra features are supported by this client. If one wants to specify multiple flags, every flag should be seperated by a unit separator (see Delimiters).

Arguments

1. Name (REQUIRED)
2. Flag (OPTIONAL)
3. ..

The possible flags for a client at this moment are the following:

- CHAT : optional feature 1 from the project manual

The flags that require additional messages have defined messages that can be found below, under a separate heading.

Example(s)

For a client without support for extra features:

`ANNOUNCE`  `Alice` 

For a client with support for the chat feature:

`ANNOUNCE`  `Alice` `CHAT`

WELCOME

The server welcomes the client. It will confirm the name of the client, and communicates the flags for the extra features that the server supports. As of now, the only supported flag by the protocol is `CHAT`. If no flags are specified (there is no second argument), the client may assume that there are no extra features supported by this server.

Arguments

1. Name (REQUIRED)
2. Flags (OPTIONAL)

The possible flags for a server at this moment are the following:

- CHAT : optional feature 1 from the project manual
- TEAMPLAY : optional feature 2 from the project manual

The flags that require additional messages have defined messages that can be found below, under a separate heading.

Example(s)

For a server without support for extra features:

`WELCOME` $_{US}$ `Alice` $_{RS}$

For a server with support for the chat feature:

`WELCOME` $_{US}$ `Alice` $_{US}$ `CHAT` $_{RS}$

REQUESTGAME

The client requests a game at the server. When the number of players is not specified, it will be set to the default value of 2. The number of players must be between 2 and 4.

Arguments

1. Number of players (OPTIONAL)

Example(s)

To request a game with two players the client MAY specify the amount of players:

`REQUESTGAME` $_{RS}$

`REQUESTGAME` $_{US}$ `2` $_{RS}$

For a game with 3 or 4 players the client MUST specify the amount of players:

`REQUESTGAME` $_{US}$ `Alice` $_{US}$ `CHAT` $_{RS}$

INFORMQUEUE

The server informs the clients about the queue for a new game.

Arguments

1. Number of players currently in queue (REQUIRED)
2. Number of players required for the game (REQUIRED)

Example(s)

If you were the first player to join a 2-player game, you'd receive the following messages:

`INFORMQUEUE` $_{US}$ `1` $_{US}$ `2` $_{RS}$

`INFORMQUEUE` $_{US}$ `2` $_{US}$ `2` $_{RS}$

STARTGAME

The server informs the clients that the game is starting. The order of players in the arguments will be the playing order, but player 1 does not necessarily start (e.g. the order could be 3-4-1-2 in a 4-player game).

Arguments

1. Name of player 1 (REQUIRED)
2. Name of player 2 (REQUIRED)
3. Name of player 3 (OPTIONAL)
4. Name of player 4 (OPTIONAL)

Example(s)

`STARTGAME` `US` `Alice` `US` `Bob` `RS`

`STARTGAME` `US` `Alice` `US` `Bob` `US` `Charlie` `RS`

`STARTGAME` `US` `Alice` `US` `Bob` `US` `Charlie` `US` `David` `RS`

NOTIFYTURN

The server informs the clients whose turn it is.

Arguments

1. 0 | 1 - Whether it is the turn of the destination client (REQUIRED)
2. Name of the player whose turn it is (REQUIRED)

Example(s)

From the perspective of Alice:

`NOTIFYTURN` `US` `0` `US` `Bob` `RS`

`NOTIFYTURN` `US` `1` `US` `Alice` `RS`

From the perspective of Bob:

`NOTIFYTURN` `US` `1` `US` `Bob` `RS`

`NOTIFYTURN` `US` `0` `US` `Alice` `RS`

MAKEMOVE

The client requesting the server to make a move on behalf of them.

Arguments

1. WORD | SWAP - The type of turn they want to do (REQUIRED)

If WORD:

2. Start coordinate, formatted as described in section ‘Coordinates’ (REQUIRED)
3. H | V - direction of the move (REQUIRED)
4. Word, where blank tiles are represented by a lowercase letter, e.g. DOg (REQUIRED)

If SWAP:

2. String of tiles they want to swap, e.g. PQG, blank tiles represented by an exclamation mark `!` (REQUIRED)

Example(s)

For placing a word:

`MAKEMOVE` `US` `WORD` `US` `C10` `US` `H` `US` `DOg` `RS`

`MAKEMOVE` `US` `WORD` `US` `A11` `US` `V` `US` `AQUA` `RS`

For swapping tiles:

`MAKEMOVE␟SWAP␟PQG␞`

`MAKEMOVE␟SWAP␟XMN!␞`

NEWTILES

The server informs the client of their newly drawn tiles after their move (or swap).

Arguments

1. String with newly drawn tiles, blank tiles represented by an exclamation mark `!` (REQUIRED)

Example(s)

After having requested 4 tiles:

`NEWTILES␟AC!A␞`

INFORMMOVE

Server informs all clients about the move that has just been made (including the client that made the move).

Arguments

1. Name of the player that made the move (REQUIRED)
2. WORD | SWAP - The type of move that has been made (REQUIRED)

If WORD:

3. Start coordinate, format examples: B3, M12 (REQUIRED)
4. H | V - direction of the move (REQUIRED)
5. Word, where blank tiles are represented by a lowercase letter, e.g. DOg (REQUIRED)

If SWAP:

3. Number of tiles the player has swapped (REQUIRED)

Example(s)

The following examples are the appropriate inform messages for the example moves in @[sec:make-move-examples]:

`INFORMMOVE␟ALICE␟WORD␟C10␟H␟DOg␞`

`INFORMMOVE␟BOB␟WORD␟A11␟V␟AQUA␞`

`INFORMMOVE␟ALICE␟SWAP␟3␞`

`INFORMMOVE␟BOB␟SWAP␟4␞`

ERROR

The server informs the client that something went wrong

Arguments

1. Error code (REQUIRED)

Example(s)

`ERROR␟E001␞`

`ERROR␟E006␞`

`ERROR␟E010␞`

GAMEOVER

The server informs clients that the game is over. If the win type is disconnect, there is no winner. The score of the disconnected player has to be set to 0 by the server.

Arguments

1. WIN | DISCONNECT - type of win (REQUIRED)
2. Player 1 name (REQUIRED)
3. Player 1 score (REQUIRED)
4. Player 2 name (REQUIRED)
5. Player 2 score (REQUIRED)
6. Player 3 name (OPTIONAL)
7. Player 3 score (OPTIONAL)
8. Player 4 name (OPTIONAL)
9. Player 4 score (OPTIONAL)

Example(s)

`INFORMMOVE␟WIN␟Alice␟257␟Bob␟332␞`

`INFORMMOVE␟WIN␟Alice␟225␟Bob␟241␟Charlie␟226␞`

`INFORMMOVE␟WIN␟Alice␟162␟Bob␟174␟Charlie␟197␟David␟200␞`

When Alice disconnected during the game:

`INFORMMOVE␟DISCONNECT␟Alice␟0␟Bob␟174␟Charlie␟197␟David␟200␞`

PLAYERDISCONNECTED

The server informs clients that a player is disconnected.

Arguments

1. Name of the disconnected player

Example(s)

`PLAYERDISCONNECTED␟Alice␞`

SENDCHAT

Client sends a chat message to the server.

Arguments

1. The chat message you want to send (REQUIRED)

Example(s)

`SENDCHAT␟HI␞`

NOTIFYCHAT

The server notifies all clients of a new chat message

Arguments

1. Name of the player that sent the message (REQUIRED)
2. The message sent (REQUIRED)

Example(s)

Errors

There are several errors that both the server and client can communicate whenever something went wrong. The errors in this protocol are as follows:

```
E001 - Name already taken
E002 - Unknown command
E003 - Invalid argument
E004 - Invalid coordinate
E005 - Word does not fit on board
E006 - Unknown word
E007 - Too few letters left in tilebag to swap
E008 - You do not have the required tiles
E009 - It is not your turn
E010 - Wrong number of players
E011 - Word not connected to other tiles
E012 - Client has already announced themselves
E013 - Client has not yet announced themselves
E014 - Client is not in a game
E015 - Game already started
```

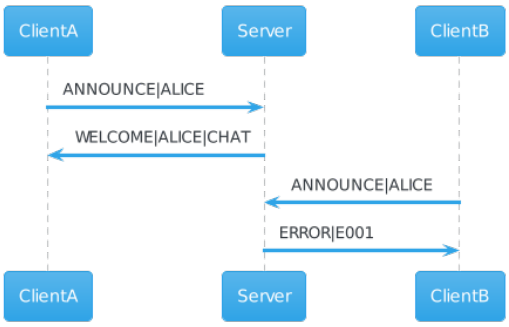
For readability, the unit separator in the sequence diagrams below are represented by `|` and the message separator is omitted (every arrow is a new message).

E001 - Name already taken

This error is sent from the server to the client as a response to an `ANNOUNCE` message if the name of the client is already taken by another client in the same server.

Example

In this example, after `ClientA`, with name `Alice` has joined the server, `ClientB` wants to do so too. Upon this request, the server replies with a 'Name already taken'-error.

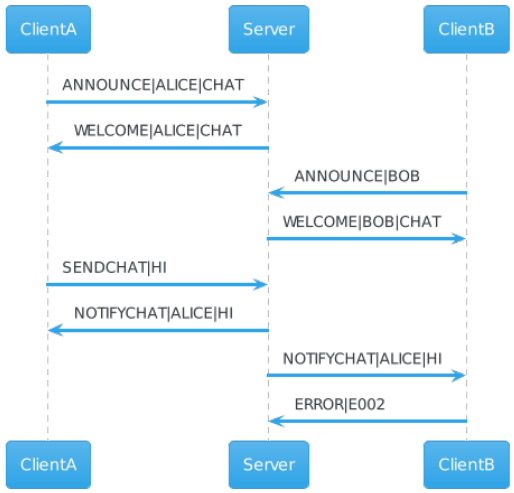


E002 - Unknown command

This error is sent from either the client to the server or from the server to the client. It is sent whenever the first unit of a message (the command) is not a known command, as defined above. This can also be the case whenever a command of one of the extra features is sent to a client/server that does not support those extra features.

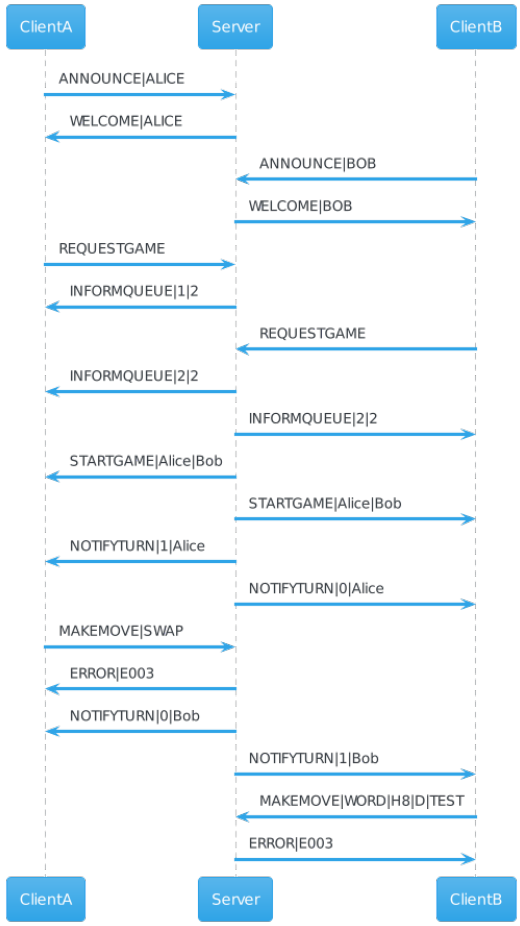
Example

In this example, the server makes a mistake by not realizing that `ClientB` does not support the extra feature chat, and hence sends a `NOTIFYCHAT` message to `ClientB`. Therefore, `ClientB` responds with an 'Unknown command'-error.



E003 - Invalid argument

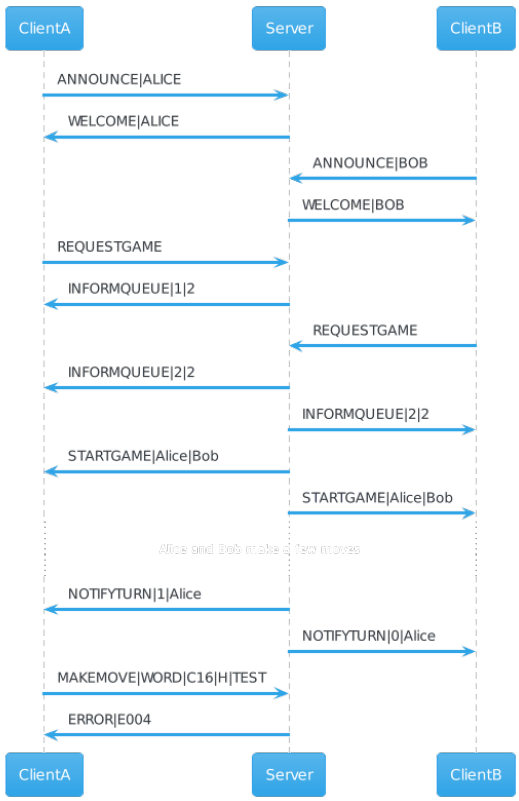
This error is sent whenever an argument is invalid. This can be whenever too few arguments are given, or when a 'closed' argument (where you have to choose from several options, such as the first argument of the `MAKEMOVE` command) is not one of the allowed options. ### Example In this example, `ClientA` wants to swap tiles, but they did not pass all the necessary arguments, namely the tiles he/she wants to swap. Therefore, `Server` responds with an 'Invalid argument'-error. For the second error, `ClientB` is trying to place a word in an invalid direction (`D`), which also triggers an 'Invalid argument'-error from the server.



E004 - Invalid coordinate

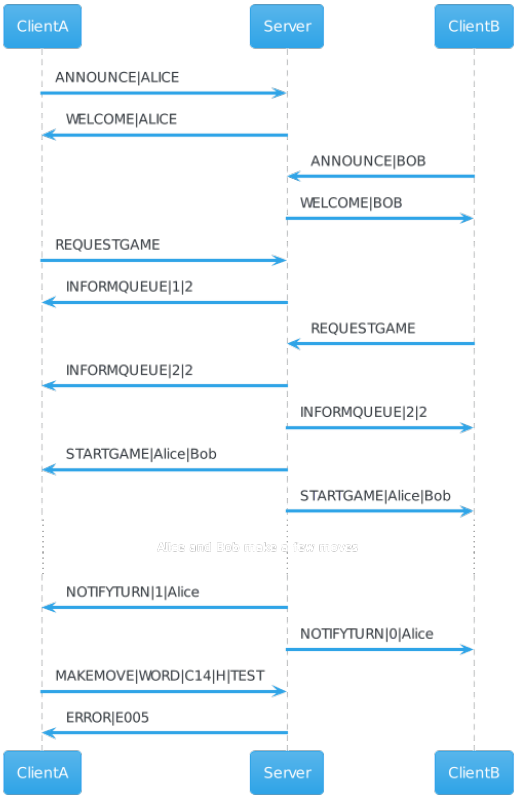
This error is sent whenever a certain coordinate is not on the board or in an invalid format. ### Example In this example, `ClientA` wants to write a word but passes a coordinate C16, which is not within the dimensions of

the board. Therefore, `Server` responds with an ‘Invalid coordinate’-error.



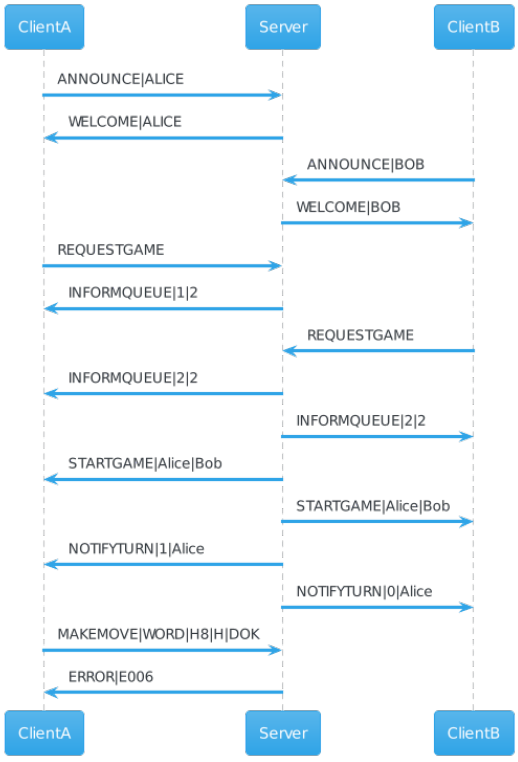
E005 - Word does not fit on board

This error is sent whenever a word of a move does not fit on the board. This can either be because the word is too long and falls off of the edges of the board, or because one or more tiles overwrite tiles that are already on the board with a different letter. ### Example In this example, `ClientA` wants to write a word but passes a coordinate C14 and writes a word which is longer than the distance to the edge of the board. Therefore, `Server` responds with a ‘Word does not fit on board’-error.



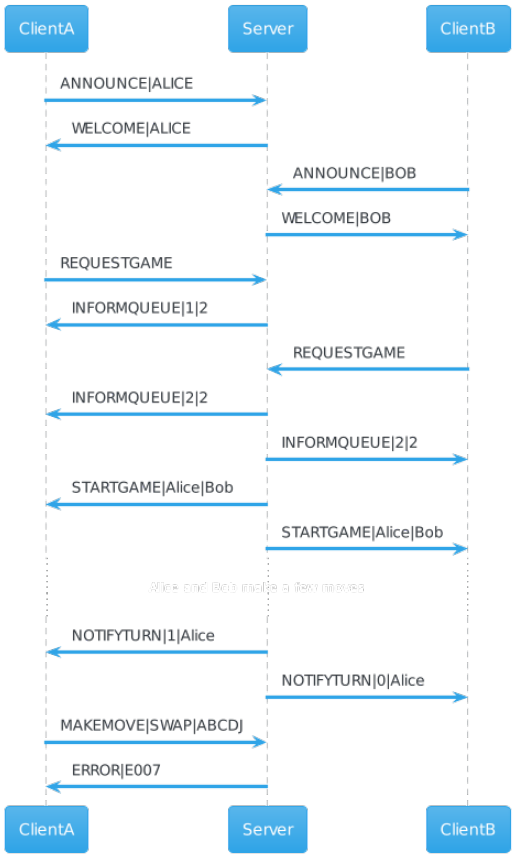
E006 - Unknown word

This error is sent whenever a word is not in the dictionary. ### Example In this example, `ClientA` wants to write the word DOK instead of a DOG or DOCK which is in fact not a valid word in the dictionary. Therefore, `Server` responds with an 'Unknown word-error'.



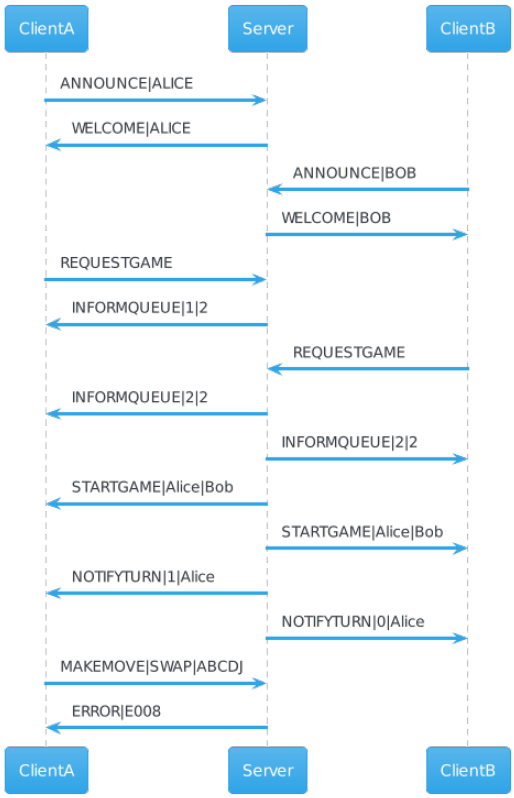
E007 - Too few letters left in tilebag

This error is sent whenever someone tries to swap more tiles than there are left in the tilebag. ### Example In this example, `ClientA` wants swap 4 letters from their rack. However, there are only 3 tiles left in the bag. Therefore, `Server` responds with a 'Too few letters left in tilebag'-error.



E008 - You do not have the required tiles

This error is sent whenever someone is trying to make a move (or swap) with tiles that he/she does not have. ### Example In this example, `ClientA` wants swap 4 letters from their rack. However, they do not have the letter J on their rack. Therefore, `Server` responds with an 'You do not have required tiles'-error.

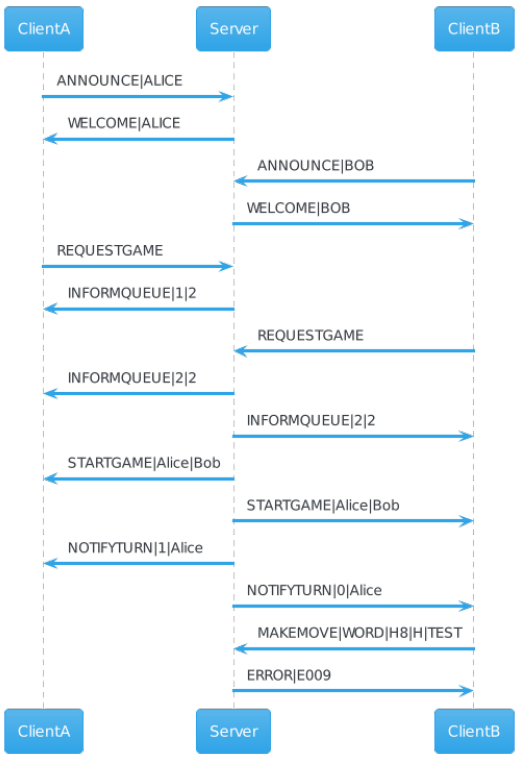


E009 - It is not your turn

This error is sent whenever someone tries to make a move before their turn is announced. Making a move when it is your turn, but before receiving your `NOTIFYTURN` message should also trigger this error.

Example

In this example, `ClientB` makes a turn when actually it's not their turn. Therefore, `Server` responds with an 'It's not your turn'-error.



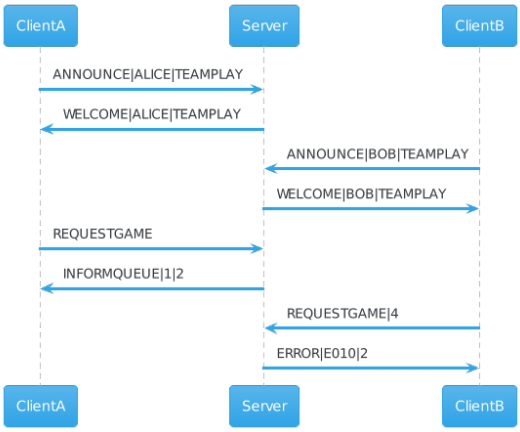
E010 - Wrong number of players game request

This error is thrown whenever - in a server that does support team-play, but does not support multi-game play - a queue for a game has already started, and another player requests a game for a different number of players. In a server that does not support team-play, the server could simply return `E003`, since any number other than 2 is not a valid argument in this case. In a server that does support multi-game play, another queue for another game can be started, and no error will be returned.

Arguments

This error will also have an argument, indicating the number of players a queue already exists for (REQUIRED).

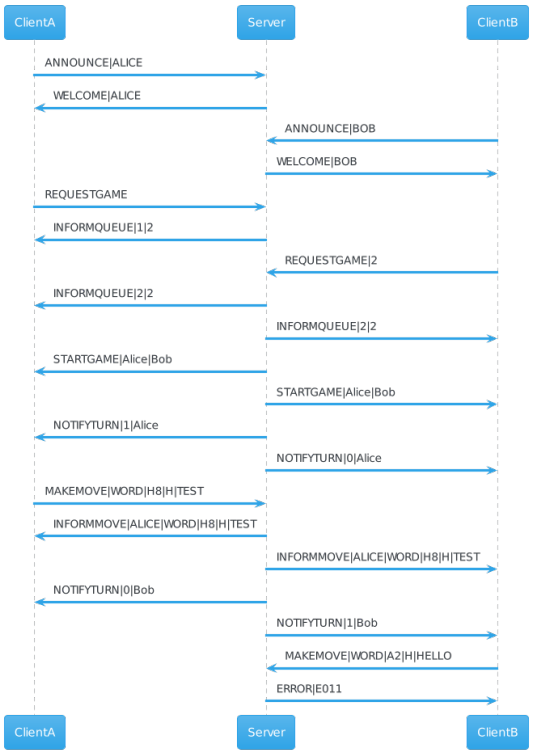
Example



E011 - Word not connected to other tiles

This error is thrown whenever someone tries to make a move where they place a word, but none of the word is not connected to any of the tiles on the board. This error is also sent whenever someone tries to place the first word on the board, but does not place any of the tiles on the center square.

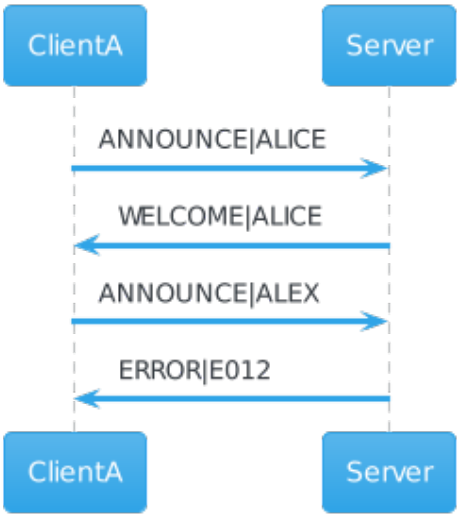
Example



E012 - Client has already announced themselves

This error is returned by the server whenever a client sends an `ANNOUNCE` message, while they have already been welcomed by the server.

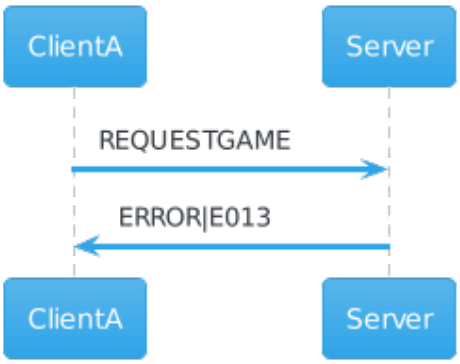
Example



E013 - Client has not yet announced themselves

This error is returned by the server whenever a client sends a `REQUESTGAME` message before they have successfully announced themselves.

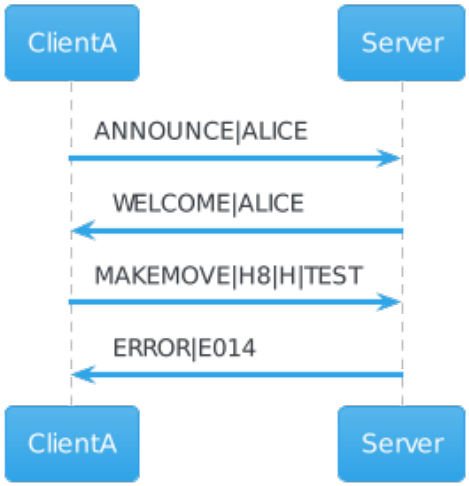
Example



E014 - Client is not in a game

This error is returned by the server whenever a client sends a game-related message, such as `MAKEMOVE` or `SWAP`, before they are in a game.

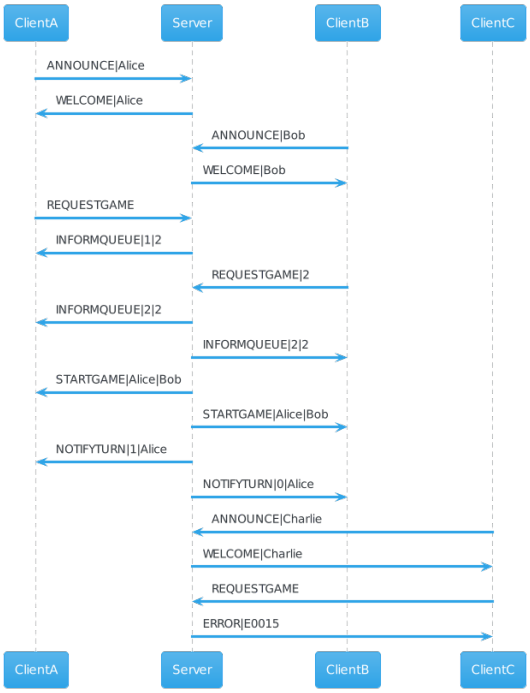
Example



E015 - Game already started

This error is returned by the server whenever a client requests a game, while a game has already started. Of course, when the server supports multi-game, this error is not necessary, and it will simply start a new queue.

Example



Happy Flow

The happy flow shows the game flow without any errors being thrown. The diagram below describes the following game summary:

- Client A joins the server, the name of client A is 'Alice', the client does not support any extra features
- Server confirms Alice that she has joined, the server does not support any extra features
- Client B joins the server, the name of client B is 'Bob', the client does not support any extra features
- Server confirms Bob that he has joined, the server does not support any extra features
- Alice requests a game for 2 players
- Server informs that there is now a queue for a 2-player game, where she is the only player waiting

- Bob requests a game for 2 players
- Server informs Alice and Bob that there is a queue for a 2-player game, where 2 players are waiting
- Server informs Alice and Bob that the game is starting, the player names are `Alice` and `Bob`
- Server informs Alice and Bob about their tiles
- Server informs Alice and Bob that it's Alice her turn
- Alice tells server that she wants to play the word `DOG`, horizontally, starting from coordinate H8
- Server informs Alice that her new tiles are `H`, `F` and `K`
- Server informs Alice and Bob that Alice has just laid down the word `DOG` horizontally, starting from H8
- Server informs Alice and Bob that it's Bob his turn
- Bob tells server that he wants to swap the tiles `P`, `Q` and `G`
- Server informs Bob that his new tiles are `A`, `C` and `A`
- Server informs Alice and Bob that Bob has swapped 3 tiles

Then, Alice and Bob keep playing for a while, after which the game ends in a win for Bob, with 332 points. Alice has scored 257 points.